GOLDMANN

computer compact

COMMODORE 64 BASIC

System und Anwendung

Richard G. Peddicord

computer compact

Richard G. Peddicord

COMMODORE 64 BASIC

System und Anwendung

Wir danken der Firma Commodore Büromaschinen GmbH für
die freundliche Überlassung des Bildmaterials.

Printed in Germany · 11/84 · 1. Auflage · 1110
© 1984 by Alfred Publishing Co. Inc.
© 1984 der deutschen Ausgabe bei Wilhelm Goldmann Verlag, München
Umschlaggestaltung: Design Team München
Übersetzung: Sabina Janas
Konzeption und Redaktion: topic GmbH, München-Karlsfeld
Hersteller: Hubert K. Hepfinger / Peter Sturm
Satz: fb Werbeservice, München
Verlagsnummer: 13116

ISBN 3-442-13116-2

INHALT

1. EINLEITUNG 7

2. DER COMMODORE 64 8

3. DAS FERNSEHGERAET ALS MONITOR 12

4. IHR ERSTES ERFOLGSERLEBNIS 14

5. ZEILENNUMMERN, KONSTANTEN UND VARIABLEN 19

6. DIE SICHERUNG VON PROGRAMMEN AUF BAND 26

7. DIE SICHERUNG VON PROGRAMMEN AUF DISKETTE 30

8. DIE BENUTZUNG EINES VIDEOMONITORS 35

9. DIE BENUTZUNG EINES DRUCKERS 37

10. PROGRAMMIEREN MIT EINGABE 41

11. EINFACHE BERECHNUNGEN 45

12. ARITHMETISCHE AUSDRUECKE 49

13. IF...THEN-ANWEISUNG 52

14. LOGISCHE AUSDRUECKE 54

15. SCHLEIFEN 56

16. SPEICHERUNG VON DATEN IN PROGRAMMEN 61

17. EIN SPESENABRECHNUNGSPROGRAMM 64

18. FELDER 67

19. DIE FORMATE
VON SCHLUESSELWOERTERN. 69

20. DIE VERSCHIEDENEN
ARTEN VON SCHLUESSELWOERTERN 72

21. LISTE DER SCHLUESSELWOERTER 75

1. Einleitung

Wenn Sie gerade einen Commodore 64 erworben haben oder dies beabsichtigen, dann ist der vorliegende Band aus der Reihe »Goldmann computer compact« gerade richtig für Sie. Er wendet sich an den Neuling, also auch an Schüler und Jugendliche, und zeigt nicht nur, wie sich der Commodore 64 an Peripheriegeräte (Zusatzgeräte), wie Videomonitor oder Fernsehgerät, Drucker, Diskettenlaufwerk oder Datasette-Rekorder, anschließen läßt. Dieser Band gibt zugleich eine Einführung in BASIC, jener Programmiersprache, die auf Ihrem Commodore 64 verwendet wird. BASIC setzt sich, wie jede andere höhere Programmiersprache, aus einer Menge von Schlüsselworten (»Keywords«) zusammen, die für den Computer jeweils eine genaue Bedeutung besitzen, sowie aus einer Reihe grammatikalischer Regeln, die festlegen, wie das jeweilige Wort zu benutzen ist. Ein Schlüsselwort, wie LIST, RUN oder PRINT, ist ein Ausdruck, den das System als Kommando oder Teil eines Kommandos erkennt; es darf also in BASIC kein Variablenname, beispielsweise zur Benennung von Dateien, verwendet werden, der eines dieser Schlüsselwörter enthält.

Um Fehler dieser Art leicht zu vermeiden, enthält dieser Band eine Liste aller BASIC-Schlüsselworte mit Beispiel, Format, Fehlermeldungen und der Beschreibung dessen, was ihre Verwendung bewirkt. Damit wird der Zugang zu BASIC wesentlich erleichtert, allerdings können Sie keinen großen Lernerfolg erwarten, wenn Sie die hier erläuterten Arbeitsschritte nicht an einem Rechner tatsächlich einüben.

2. Der Commodore 64

Mit dem Commodore 64 haben Sie einen der leistungsfähigsten Personal Computer erworben, die sich derzeit auf dem Markt befinden. Zusammen mit dem Rechner erhalten Sie ein Bedienungshandbuch, das die wichtigsten Informationen enthält, die zur Inbetriebnahme des Geräts erforderlich sind, sowie ein Netzgerät. Es versorgt Ihren Commodore 64 mit der nötigen Betriebsspannung und ist nicht in den Rechner selbst integriert, um dessen Hitzebelastung zu vermindern und sein Gewicht zu reduzieren. Neben dem Stromversorgungskabel weist das Netzgerät ein zweites Kabel auf, an dessen Ende sich ein Stecker mit sieben kleinen Stiften und einer Schutzverkleidung befindet. Damit wird das Netzgerät an den Commodore 64 angeschlossen; die entsprechende Buchse finden Sie an der rechten Seite des Geräts. Stellt man ein System zusammen, so arbeitet man immer nur an einer Komponente. Auf diese Weise lassen sich eventuelle Fehler unter Kontrolle halten. Jetzt ist es an der Zeit zu überprüfen, ob Ihr Commodore 64 mit Strom versorgt wird.

Suchen Sie den kleinen Ein/Aus(On/Off)-Schalter neben der Buchse für den Anschluß des Netzgeräts. Wenn Sie ihn betätigen, sollte die rote Anzeige (POWER) oberhalb der Tastatur aufleuchten. Ist das der Fall, können Sie sich gratulieren und weitermachen.

Brennt diese Anzeige dagegen nicht, beginnen Sie nochmals von vorne.

Drehen Sie nun den Commodore 64 herum, so daß Sie vor der Rückseite des Geräts sitzen. Der Schlitz links dient der Aufnahme von ROM-Chips und wird Modul-Steckplatz genannt. Es gibt zwei Arten von Speicher, »Read Only Memory«, kurz ROM genannt, und »Random Access Memory«, abgekürzt RAM. Während man im RAM Informa-

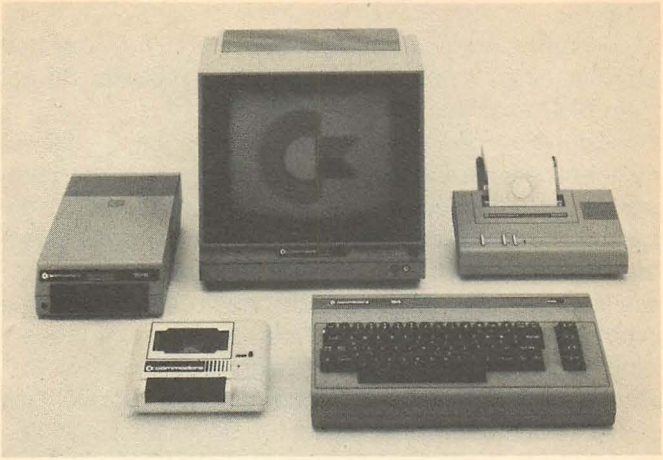


Abb. 1: Eine typische Systemkonfiguration: Commodore 64, 1701 Monitor, 1520 Drucker, 1541 Diskettenlaufwerk, Datassette-Rekorder.

tion beliebig speichern und wieder löschen kann, läßt sie sich aus einem ROM nur lesen; die meisten ROM-Chips enthalten Programme für bestimmte Anwendungsbereiche Ihres Computers. Der Vorteil dieser ROM-Chips besteht darin, daß die darauf gespeicherten Programme keinen Platz im Hauptspeicher des Commodore 64 beanspruchen und auch nicht von einem Diskettenlaufwerk oder Kassettenrekorder hineingelesen werden müssen. Es ist sogar möglich, mehr als einen ROM-Chip einzustecken.

Neben dem Schacht für die ROM-Chips befindet sich ein Schalter, womit Sie den Fernsehkanal wählen, auf den Sie die Computer-Ausgabe leiten möchten. Es stehen Ihnen Kanal 3 und 4 zur Verfügung. Probieren Sie beide auf Ihrem Fernsehgerät aus, um zu sehen, welcher Kanal das bessere Bild liefert.

Direkt neben diesem Wahlschalter liegt der TV-Anschluß. Das Ihrem Commodore 64 beigelegte Hi-Fi-Kabel läßt sich hier anschließen; das andere Ende verbinden Sie mit dem UHF-Anschluß Ihres Fernsehers.
Besitzen Sie einen Videomonitor, dann legen Sie das Hi-Fi-Kabel beiseite. Ein Videomonitor sieht aus wie ein Fernsehgerät, verfügt jedoch über kein Empfangsgerät für die verschiedenen Fernsehkanäle. Er akzeptiert nur ein Videosignal, wie es auch eine Videokamera erzeugt.
Ein Videomonitor läßt sich am Audio/Video-Ausgang rechts vom TV-Anschluß mit dem Computer verbinden; eine genaue Anleitung finden Sie in Kapitel 8.
Rechts neben dem Audio/Video-Ausgang befindet sich die Anschlußmöglichkeit für serielle Schnittstellen (»Interface«). Hier schließen Sie das Diskettenlaufwerk, den Drucker oder jedes andere Gerät an, das für serielle Datenübertragung geeignet ist. Damit ist gemeint, daß die Datenübertragung Bit für Bit erfolgt, im Gegensatz zu einer parallelen Schnittstelle, die jeweils ein ganzes Byte (8 Bit) übermittelt. Mit dem Begriff »Schnittstelle« ist, allgemein gesprochen, lediglich die Verbindung zweier Systemkomponenten gemeint. Rechts von diesem seriellen Anschluß finden Sie die Anschlußmöglichkeit für den Datassette-Rekorder oder ähnliche Kassettenrekorder. Ganz rechts auf der Rückseite des Commodore 64 liegt ein frei programmierbarer Ein- und Ausgang (»User Port«), der ebenfalls als Anschlußmöglichkeit für Schnittstellen genutzt werden kann.
Gleich vorweg ein Wort zu Schnittstellen: die Unterscheidung zwischen paralleler und serieller Übertragung reicht noch nicht aus, um sicher festzulegen, daß in jedem Fall damit die Übertragung zwischen Computer und Drucker beispielsweise gelingt. Es existieren zahlreiche Varianten der verschiedensten Hersteller, und Sie sollten bei einem Kauf absolut sichergehen, daß die gewählte Schnittstelle mit den Komponenten Ihres Systems verträglich ist. Prü-

fen Sie dies daher eingehend. Auf dem Gebiet der seriellen Schnittstellen ist die »RS232 C«-Schnittstelle sehr verbreitet, die in Europa auch unter der Modellbezeichnung »V 24« angeboten wird. Problemen, die hier entstehen können, gehen Sie am leichtesten dadurch aus dem Weg, daß Sie die von Commodore hergestellten Peripheriegeräte zur Erweiterung Ihres Systems erwerben. Dann ist garantiert, daß Sie nicht Geld und Nerven vergeuden müssen, um die Erweiterungskomponenten so arbeiten zu lassen, wie Sie es sich vorstellen.

3. Das Fernsehgerät als Monitor

Ihr Commodore 64 läßt sich auch an ein gewöhnliches Fernsehgerät anschließen. Einzelheiten dazu finden Sie im beigelegten Bedienungshandbuch. Nehmen Sie diesen Anschluß gleich nach Erwerb des Geräts vor, um sicherzustellen, daß der RF-Modulator funktioniert. Er sendet die separaten Video- und Audiosignale an den Fernsehkanal 3 oder 4. Um allerdings die Möglichkeiten auszuschöpfen, die Ihr Commodore 64 an Farbgraphik bietet, benötigen Sie den Commodore 1701 Farbmonitor oder ein vergleichbares Gerät.
Schalten Sie zuerst das Fernsehgerät, dann Ihren Commodore 64 ein. Auf dem Bildschirm sollte, in hellblauer Schrift auf dunkelblauem Hintergrund, folgende Meldung erscheinen:

```
    **** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM 38911 BASIC BYTES FREE

READY
```

Direkt unter dem Wort READY blinkt ein hellblaues Rechteck, der Kursor. Er zeigt nicht nur an, daß das Gerät auf Ihre Eingabe wartet, sondern markiert zugleich die Stelle, an der das Zeichen plaziert wird, das Sie als nächstes auf der Tastatur eintippen.
Bleibt Ihr Bildschirm jedoch leer, so tun Sie folgendes:

1. Überprüfen Sie, ob die Geräte richtig verbunden sind, und ob die Anschlüsse festsitzen.
2. Überprüfen Sie die Einstellung Ihres Fernsehgeräts; Sie müssen Kanal 3 oder 4 wählen.

3. Kontrollieren Sie die Stellung des Kanalwahlschalters auf der Rückseite Ihres Commodore 64. Er muß nach links zeigen, wenn Sie die Rückseite des Geräts betrachten.
4. Besitzen Sie einen 1701 Monitor, kontrollieren Sie die Schalterstellung auf der Rückseite.
5. Studieren Sie nochmals die Anleitungen des Bedienungshandbuchs.
6. Wenden Sie sich mit ungelösten Problemen an Ihren Händler.

4. Ihr erstes Erfolgserlebnis

Das blinkende Rechteck ist der Kursor, der, wie gesagt, die Stelle markiert, an der das von Ihnen auf der Tastatur eingegebene Zeichen auf dem Bildschirm erscheint. Tippen Sie einige Zeichen und beobachten Sie, wie sich der Kursor dabei weiterbewegt. Geben Sie dann etwas völlig Unverständliches ein und betätigen Sie die RETURN-Taste, die sich ganz rechts auf der Tastatur befindet. Das System wird mit folgender Ausgabe antworten:

```
?SYNTAX ERROR
```

Auf diese Weise teilt es Ihnen mit, daß es Sie nicht verstanden hat. Es gibt aber auch noch andere Formen der Fehlermeldung.
Spielen Sie mit dem Kursor und lassen Sie ihn über den Bildschirm wandern. Benutzen Sie die SHIFT-Taste sowie die Kursortasten für die senkrechte und waagrechte Bewegung des Kursors.
Wenn Ihnen die Spielerei zu langweilig wird und Sie ernsthaft arbeiten wollen, dann drücken Sie die RETURN-Taste und geben anschließend die folgende Zeile ein:

```
PRINT "DIE SONNE GEHT IM OSTEN AUF,"
```

Unterläuft Ihnen beim Tippen ein Fehler, und entdecken Sie ihn sofort, dann benutzen Sie die INST/DEL-Taste (oben rechts), um an die betreffende Stelle zurückzugehen und von dort aus weiterzumachen. Oder Sie

löschen die ganze Zeile, indem Sie die RETURN-Taste drücken und eine neue Zeile anfangen.
Wenn Ihre Zeile zu stimmen scheint, bedienen Sie die RETURN-Taste. Sie sollten dann folgende Ausgabe auf dem Bildschirm erhalten:

```
DIE SONNE GEHT IM OSTEN AUF,
READY
```

So weit, so gut. Wenn Sie dabei Probleme hatten, dann schalten Sie das Gerät aus und wieder an. Versuchen Sie etwas anderes:

```
PRINT 3*15
45
```

Geben Sie nun das Kommando LIST ein und bedienen Sie dann die RETURN-Taste. Sie erhalten nur eine leere Zeile. Es gibt kein Programm, das aufgelistet werden könnte, weil Sie keine Zahl vor der PRINT-Anweisung angegeben haben. Sie befanden sich deshalb in der Betriebsart der direkten Ausführung, bei der jedes Kommando unmittelbar nach dem Drücken der RETURN-Taste sofort ausgeführt wird.
Geben Sie nun die folgende Anweisung mit einer Zeilennummer ein:

```
10 PRINT "DIE SONNE GEHT IM OSTEN AUF,"
```

Drücken Sie jetzt die RETURN-Taste, dann geschieht nichts, außer daß der Kursor in die nächste Zeile wandert. Das System hat jedoch diese Anweisung als Teil Ihres sich

entwickelnden Programms gespeichert, weil Sie eine Anweisungsnummer (auch Zeilennummer genannt) vorangestellt haben. Alle Anweisungen werden nach aufsteigenden Zeilennummern angeordnet. Wenn Ihnen beim Tippen der Zeile ein Fehler unterläuft, können Sie die in der oberen rechten Ecke der Tastatur befindliche INST/DEL-Taste zur Korrektur benutzen. Wenn Sie SHIFT und INST/DEL drücken, dann schaffen Sie an der Position des Kursors Platz für ein Zeichen, und wenn Sie INST/DEL allein bedienen, dann bewegt sich der Kursor auf die Position des zuletzt eingegebenen Zeichens zurück.
Eine Warnung: Wenn Sie sich innerhalb der Anführungszeichen befinden, dann arbeiten die Kursortasten nicht in dieser Weise. Wenn Sie sie verwenden wollen, geht das nur nach Eingabe eines zweiten Anführungszeichens.
Versuchen Sie nun, die restlichen drei Zeilen dieses Programms einzugeben, und vergessen Sie dabei nicht, nach jeder Zeile die RETURN-Taste zu bedienen:

```
20 PRINT "DER OSTERHAS BEGINNT DEN LAUF."
30 PRINT "UM EINEN KORB VOLL EIER SITZEN"
40 PRINT "DREI HAESLEIN, DIE DIE OHREN SPITZEN."
```

Geben Sie das Kommando RUN ein und drücken Sie RETURN. Dann sollte folgendes auf Ihrem Bildschirm erscheinen:

```
DIE SONNE GEHT IM OSTEN AUF,
DER OSTERHAS BEGINNT DEN LAUF.
UM EINEN KORB VOLL EIER SITZEN
DREI HAESLEIN, DIE DIE OHREN SPITZEN.
```

Sie sehen, daß die Ausdrücke in Anführungszeichen genau so gedruckt werden, wie Sie sie eingegeben haben,

allerdings ohne Anführungszeichen. Der Autor dieser Zeilen ist übrigens Christian Morgenstern. Tippen Sie nun LIST ein und drücken Sie RETURN. Diesmal wird Ihr Vierzeilenprogramm ausgegeben. Wenn jemand vor Ihnen den Rechner benutzt hat und das Gerät zwischenzeitlich nicht ausgeschaltet wurde, erhalten Sie wahrscheinlich auch noch andere Ausgaben.
Um ein solches noch im Hauptspeicher stehendes Programm zu löschen, müssen Sie nur das Kommando NEW eingeben und die RETURN-Taste drücken. Versuchen Sie es doch gleich einmal. Wenn Sie anschließend LIST eintippen, wird kein Programm mehr erscheinen.
Jedesmal also, wenn Sie ein neues Programm beginnen wollen, müssen Sie NEW eingeben und die RETURN-Taste betätigen.

Übungen

1. Arbeiten Sie mit direkter Ausführung (also ohne Zeilennummern) und lassen Sie das System Ihren Namen ausgeben.
2. Schreiben Sie ein vierzeiliges Programm, das das folgende Gedicht von Franz Pocci ausgibt:

```
DER DRACHEN SCHAUT, IHR LIEBEN LEUT,
DER NICHTS ALS FEUER UND FLAMMEN SPEIT;
DA KOMMT EIN RITTERSMANN DAHER
UND STICHT IHN TOT MIT SEINEM SPEER.
```

3. Geben Sie folgende Anweisung ein:

```
PRINT "8+8"
```

Wie reagiert der Computer darauf, wenn Sie nun die RETURN-Taste bedienen?
Versuchen Sie dagegen folgende Eingabe:

```
PRINT 8+8
```

4. Schreiben Sie ein Programm, das folgende Liste ausgibt:

```
ACCESS: ZUGRIFF
TO ALTER: AENDERN
BIT: BINARY DIGIT
BLANK: LEERSTELLE
BRACKET: KLAMMER
BRANCH: VERZWEIGUNG
BUS: SAMMELLEITUNG
```

5. Zeilennummern, Konstanten und Variablen

Zeilennummern zeigen dem Computer an, daß die numerierten Anweisungen Bestandteil eines Programms sind und im Hauptspeicher abgelegt werden sollen. Zeilennummern sind aber auch noch für andere Dinge gut. Sie legen nämlich die Reihenfolge fest, in der der Computer Ihre Anweisungen ausführt. Außerdem listet er gleichzeitig Ihr Programm in aufsteigender Folge der Zeilennummern auf.
Geben Sie beispielsweise folgendes Programm ein:

```
NEW
20 PRINT "SPART ALTERSHEIM"
10 PRINT "JUNG KAPUTT"
```

Lassen Sie dieses Programm mit Hilfe von LIST ausgeben, erscheint es, in aufsteigender Folge der Zeilennummern geordnet, auf dem Bildschirm:

```
10 PRINT "JUNG KAPUTT"
20 PRINT "SPART ALTERSHEIM"
```

Zeilen werden gewöhnlich mit einem Vielfachen der Zahl 10 durchnumeriert, damit die Einfügung weiterer Zeilen, die zur Korrektur oder Modifizierung eines bereits existierenden Programms erforderlich sein können, ohne weiteres möglich ist. Zeilennummern können alle positiven ganzen Zahlen zwischen 1 und 6500 sein.
Eine Zeile eines Programms wird gelöscht, indem man die entsprechende Zeilennummer eintippt und unmittelbar anschließend die RETURN-Taste anschlägt. Versuchen Sie folgende Eingabe:

Und jetzt lassen Sie mit Hilfe von LIST das Programm auflisten. Die Ausgabe müßte wie folgt aussehen:

```
20 PRINT "SPART ALTERSHEIM"
```

Zeilennummern dienen dazu, die einzelnen Anweisungen eines Programms zu markieren. Durch sie kann der Computer jede Zeile des Programms wieder auffinden. Auf diese Weise kann man auch den Teil des Programms, den man mit Hilfe des Kommandos LIST auflisten möchte, genau angeben.

LIST	Auflisten aller Zeilen
LIST 10	Auflisten von Zeile 10
LIST 10-	Auflisten von Zeile 10 bis zum Ende
LIST -30	Auflisten vom Anfang bis zu Zeile 30
LIST 10-30	Auflisten von Zeile 10 bis Zeile 30

Konstanten sind Werte, die man Variablen zuordnet, wie man beispielsweise an folgendem Programmstück sehen kann:

```
NEW
10 LET PI=3.14159
20 PRINT PI
RUN
3.14159

READY
```

Eine Anweisung, wie sie Zeile 10 darstellt, wird Zuweisungs- oder LET-Anweisung genannt. Das Schlüsselwort LET kann man dabei auch weglassen. Bei der Ausführung einer Zuweisung wird der Wert auf der rechten Seite des

Gleichheitszeichens in derjenigen Speicherzelle abgelegt, die dem Variablennamen links vom Gleichheitszeichen zugeordnet ist.
Nimmt man später wieder auf den Variablennamen π Bezug, dann liefert das den Wert 3.14159, es sei denn, der Wert ist inzwischen geändert worden.
Konstanten können auch Wörter sein, wie beispielsweise in folgendem Programmstück:

```
NEW
10 N$="PANIK"
20 PRINT "KEINE"; N$
RUN
KEINE PANIK
```

Beachten Sie, daß der Variablenname, der die Zeichenreihe PANIK als Wert hat, auf ein Dollarzeichen ($) endet. Dadurch wird dem System angezeigt, daß in der entsprechenden Speicherzelle eine Zeichenreihe und nicht eine Zahl abgespeichert werden soll.
Die Zeichenreihe KEINE in Zeile 20 ist in Anführungszeichen eingeschlossen; dadurch weiß die PRINT-Routine, daß sie diese Zeichenreihe genau so ausgeben soll, wie sie dasteht. An dem Semikolon (;) erkennt die PRINT-Routine, daß sie keine zusätzlichen Leerzeichen vor dem nächsten auszudruckenden Element einfügen soll. Da N$ nicht in Anführungszeichen eingeschlossen ist, erscheint als Ausgabe das, was in der entsprechenden Speicherzelle abgelegt ist. (Eine Routine ist ein Bestandteil eines Programms und erledigt bestimmte Aufgaben, wie beispielsweise die Ausgabe eines Zeichens auf dem Bildschirm.)
Bei der Wahl der Namen Ihrer Variablen haben Sie weitgehende Freiheit, allerdings müssen Sie folgende Regeln einhalten:

1. Das erste Zeichen eines Variablennamens muß ein Buchstabe sein.
2. Das zweite Zeichen kann ein Buchstabe oder eine Ziffer sein.
3. Als folgende Zeichen können Sie Buchstaben oder Ziffern wählen, allerdings werden sie vom System nicht beachtet.
4. Ein Variablenname darf keines der Schlüsselwörter von BASIC enthalten (wie beispielsweise PRINT, RUN oder NEW).
5. Das letzte Zeichen des Namens muß bei Stringvariablen das Dollarzeichen ($) und bei ganzzahligen Variablen das Prozentzeichen (%) sein. Als String-Variable bezeichnet man eine Variable, die eine Zeichenfolge speichert (Buchstaben und auch Ziffern, aber keine Zahlen).
6. Ein Variablenname darf nicht unflätig sein.

Gut. Ehe Sie zum nächsten Abschnitt übergehen, ein letztes Beispiel. Geben Sie das folgende Programm ein und lassen Sie es laufen:

```
NEW
10 REM LEBENSWEISHEIT
20 BE$="UEB"
30 FU$="IMMER"
40 BV$="TREU UND"
50 LO$="REDLICHKEIT"
60 ?BE$;BE$:?FU$:?BV$:?LO$
RUN

UEB
IMMER
TREU UND
REDLICHKEIT

READY
```

Das Fragezeichen (?) stellt eine Kurzform des Kommandos PRINT dar. Überall dort, wo Sie das Wort PRINT in einem Programm sehen, können Sie stattdessen auch ein Fragezeichen verwenden. Beachten Sie, daß in Zeile 50 eine einzige Anweisung insgesamt vier PRINT-Befehle enthält. Jedes dieser Kommandos muß dabei vom nächsten durch einen Doppelpunkt (:) getrennt werden.

Übungen:

1. Geben Sie folgendes Programm ein und lassen Sie es laufen.

```
10 PRINT "OHNE"
20 PRINT "FLEISS"
30 PRINT "KEIN"
40 PRINT "PREIS"
```

2. Lassen Sie zunächst alle vier Zeilen auflisten, dann nur Zeile 30, dann die Zeilen 20 und 30 zusammen.
3. Lassen Sie alle vier Zeilen auflisten und ändern Sie alle Zeilen in folgender Weise ab: Fügen Sie jeweils vor dem abschließenden Anführungszeichen ein Leerzeichen ein und nach dem abschließenden Anführungszeichen ein Semikolon; aus

```
10 PRINT "OHNE"
```

   wird damit also:

```
10 PRINT "OHNE ";
```

   Lassen Sie das so modifizierte Programm laufen; wenn Sie alles richtig gemacht haben, steht der Spruch jetzt in einer einzigen Zeile.
4. Versuchen Sie, die Anweisung NORBERT=2 direkt ausführen zu lassen. Was erhalten Sie? SYNTAX

ERROR, weil in dem Variablennamen NORBERT ein Schlüsselwort versteckt ist. Schauen Sie in Kapitel 21 nach, um es zu erkennen.

5. Lassen Sie die folgende Zeile direkt ausführen:

```
A=5.5 : PRINT A
```

Das sollte gelingen. Jetzt versuchen Sie es einmal mit der folgenden Zeile:

```
A%=5.5 : PRINT A
```

Können Sie sich vorstellen, was passiert ist?

6. Überlegen Sie sich zuerst, was durch das folgende Programm ausgegeben wird, und lassen Sie es anschließend laufen. Haben Sie recht gehabt?

```
10 ABACUS=1
20 ABRAHAM=2
30 PRINT ABACUS
```

7. Überlegen Sie sich, was durch das folgende Programm ausgegeben wird, und lassen Sie es anschließend laufen.

```
A$="BOO "
20 B$="HOO "
30 ?A$ B$ A$ B$ : ? B$ B$
```

8. Was erbringt das Kommando LIST jeweils im folgenden Programm? Testen Sie es!

```
NEW
LIST
10 B=2
70 ?B

60 B=B+1
B=5
LIST
```

9. Geben Sie folgendes Programm ein:

```
10 R1$="EINNAHMEN"
20 R2$="AUSGABEN":K=0.25*E
30 R3$="GEWINN":G=E-A
40 ?R1$;E:?R2$;K:?:?R3$;G
```

Welche Veränderungen müssen Sie vornehmen, um die Zahlen jeweils in einer einzelnen Zeile zu erhalten?

6. Die Sicherung von Programmen auf Band

Commodore stellt zwei verschiedene Kassettenrekorder her, die sich mit dem Kassettenanschluß Ihres Commodore 64 (den kleinsten der rechteckigen Anschlüsse auf der Rückseite) direkt verbinden lassen. Den Strom erhalten diese Geräte vom Computer.
Es gibt zum einen die VIC Datassette, ein älterer, größerer, für den VIC 20 entworfener Rekorder, und zum anderen eine neuere, kompaktere Datassette, die für den C64 gedacht ist. Sie arbeiten beide in der gleichen Art und Weise. Stecken Sie das Verbindungskabel des Rekorders in den Kassettenanschluß des C64. Es handelt sich, wenn Sie von vorne auf Ihren Commodore schauen, um die zweite Buchse von links.

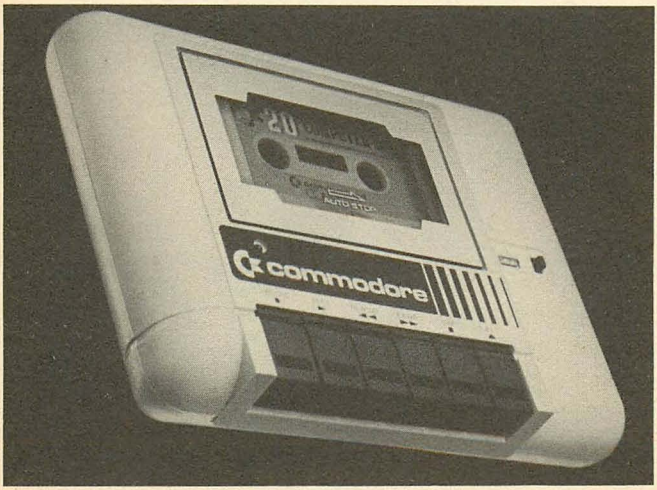


Abb. 2: Der Datassette-Rekorder

Achten Sie darauf, daß Sie das Kabel korrekt einstecken und wenden Sie keine Gewalt an; es gelingt schon, wenn Sie behutsam vorgehen.
Der Datassette-Rekorder arbeitet wie jeder andere Kassettenrekorder auch und verwendet dieselbe Art von Magnetband. Probieren Sie das Einlegen und Herausnehmen einer Kassette, damit Sie darin Übung bekommen. Die STOP/EJECT-Taste öffnet den Kassettenschacht. F.FWD (»Fast Forward«) und REW (»Rewind«) lassen das Band vorwärts und rückwärts laufen. PLAY und REC (»Record«) tun genau das, was Sie davon erwarten.
Wenn Ihr Datassette-Rekorder erst einmal angeschlossen ist, und Sie eine leere Kassette eingelegt haben, dann können Sie Ihre BASIC-Programme sichern. Geben Sie dazu ein Programm ein, und lassen Sie es laufen. Tippen Sie dann:

```
SAVE "<programmname>"
```

Ihr <programmname> kann bis zu 16 Zeichen lang sein. Der Name befindet sich im Kommando selbst in Anführungszeichen.
Wenn Sie die RETURN-Taste drücken, dann antwortet der Computer mit:

```
PRESS PLAY AND RECORD ON TAPE
```

Drücken Sie also sowohl die RECord- als auch die PLAY-Taste auf dem Datassette-Rekorder. Sobald die Sicherung erfolgt ist, erscheint das READY-Promptzeichen.
Zum Laden des gesicherten Programms vom Datassette-Rekorder spulen Sie das Band an den Anfang zurück und geben folgendes ein:

```
LOAD "PROGRAMMNAME"
```

Nach dem Drücken der RETURN-Taste zeigt der Bildschirm:

```
PRESS PLAY ON TAPE
```

Diese Meldung verschwindet, sobald der Datassette-Rekorder läuft; wenn Ihre Datei gefunden worden ist, erhalten Sie eine entsprechende Meldung. Drücken Sie dann die Commodore-Taste, und das Programm wird geladen.
Wenn Sie beabsichtigen, auf einer Seite eines Magnetbands mehrere Programme zu sichern, dann gehen Sie in folgenden Schritten vor:

1. Legen Sie die Kassette ein, und spulen Sie das Band auf den Anfang zurück.
2. Drücken Sie den Knopf neben dem Bandzählwerk, um den Zähler auf 000 einzustellen.
3. Sichern Sie die Datei wie oben erläutert.
4. Lassen Sie das Band um etwa zehn Einheiten weiterlaufen.
5. Merken Sie sich den Stand des Zählwerks und den Namen Ihrer nächsten Datei. Dadurch ist deren Anfang gekennzeichnet.
6. Sichern Sie die Datei.

Um Ihre gesicherten Dateien zu laden, gehen Sie folgendermaßen vor:

1. Legen sie die Kassette ein und spulen Sie das Band an den Anfang zurück.
2. Drücken Sie den Knopf neben dem Bandzählwerk, um den Zähler auf 000 einzustellen.
3. Lassen Sie das Band durch Drücken der F.FDW-Taste bis kurz vor dem Zählerstand laufen, an dem sich das gewünschte Programm befindet.
4. Tippen Sie LOAD am Computer ein und drücken Sie die RETURN-Taste.
5. Drücken Sie die PLAY-Taste des Datasette-Rekorders, wenn der Computer Sie dazu auffordert.

6. Nachdem der Computer das Programm gefunden und Ihnen den Namen mitgeteilt hat (während des Suchvorgangs ist der Bildschirm hellblau), drücken Sie die C=-Taste im linken unteren Teil des Tastenfeldes.
7. Ist das Programm geladen (Sie erhalten eine READY-Meldung auf dem Monitorbildschirm), betätigen Sie die STOP-Taste des Datasette-Rekorders.
8. Tippen Sie RUN ein und drücken Sie die Return-Taste, um das Programm zu starten.

Falls sich Ihre Datei nicht laden läßt, kann es am falschen Zählerstand liegen. Das passiert oft, wenn zwei verschiedene Rekorder zur Sicherung und zum Laden der Datei verwendet wurden.

7. Die Sicherung von Programmen auf Diskette

Früher oder später werden Sie ein Diskettenlaufwerk an den Commodore 64 anschließen wollen. Das Modell 1541 ist das für den C64 konzipierte Laufwerk. Es verfügt über einen eigenen Mikroprozessor, weshalb es ohne Beanspruchung der Speicher des Commodore C64 arbeitet. Eine Diskette ist ein angenehmeres und effizienteres Speichermedium als das Magnetband, allerdings ist sie auch teurer und empfindlicher.

Beim Auspacken des Diskettenlaufwerks finden Sie zusätzlich zum Stromkabel noch ein kurzes schwarzes Kabel, wodurch das Laufwerk mit Ihrem Commodore 64 verbunden wird. Das eine Ende läßt sich in den seriellen Anschluß (rechts neben dem Audio/Video-Anschluß, wenn Sie den Rechner von hinten betrachten) einstecken, und das andere Ende in einen der beiden identischen Anschlüsse auf der Rückseite des Diskettenlaufwerks.

Das Modell 1541 eignet sich für 5¼-Zoll-Disketten, die soft-sektoriert sind und ein- oder beidseitig mit einfacher oder doppelter Dichte aufzeichnen. Die Diskette selbst ist nichts anderes als eine sehr dünne Plastikscheibe, beschichtet mit magnetisierbarem Material. Sie dreht sich in ihrer schützenden Hülle.

Im Inneren des Diskettenlaufwerks befindet sich ein Lese/Schreibkopf, der winzige Felder, sogenannte Spuren und Sektoren, auf der Diskette magnetisieren oder diese Magnetisierung wieder von der Diskette lesen kann. Er bewegt sich hin und her, um auf die verschiedenen Bereiche der Diskette zuzugreifen. Dadurch wird das Geräusch verursacht, das Sie beim Betrieb des Diskettenlaufwerks hören.

Das Diskettenlaufwerk 1541 kann entweder einseitig oder beidseitig aufzeichnende Platten verwenden. Beidseitig

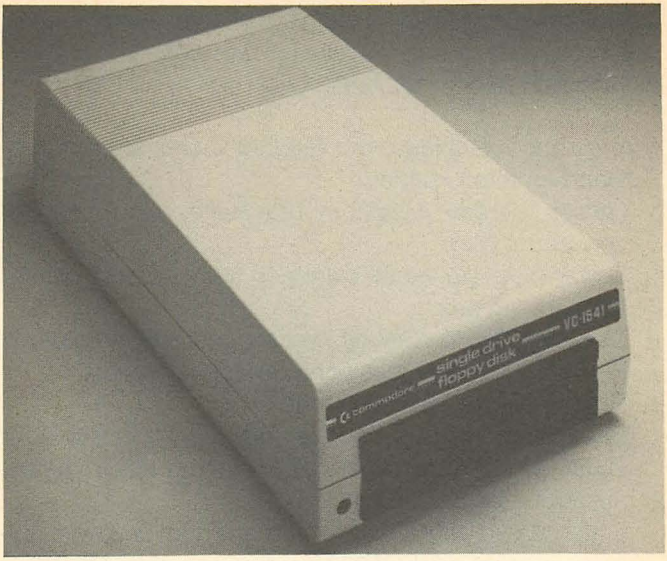


Abb. 3: Das 1541 Diskettenlaufwerk

aufzeichnende Disketten können doppelt so viel Information aufnehmen wie einseitig aufzeichnende, kosten aber auch mehr Geld.
Die Dichte der Diskette besagt, wie dicht die Information darauf gespeichert werden kann. Disketten mit doppelter Dichte sind von höherer Qualität als Disketten mit einfacher Dichte, aber das Diskettenlaufwerk 1541 ist nicht so konstruiert, daß es von der erhöhten Informationskapazität profitieren kann.
Der Begriff »soft-sektoriert« gibt an, auf welche Weise das Diskettenlaufwerk die Information auf der Platte kennzeichnet. Es hinterläßt magnetische Signale auf der Diskette, um die Daten wieder auffinden zu können. Einige Disketten weisen zu diesem Zweck auch eingestanzte Löcher auf; man bezeichnet sie dann als hart-sektoriert.

Diese Diskette ist für Ihr System am besten geeignet: Eine soft-sektorierte 5¼-Zoll-Diskette, die einseitig und mit einfacher Dichte aufzeichnet.

Hier einige Vorschläge, wie Sie Ihre Disketten in gutem Zustand erhalten können:

1. Lassen Sie die Diskette in ihrer Hülle, wenn Sie nicht verwendet wird.
2. Setzen Sie sie nicht extremer Hitze oder Sonnenbestrahlung aus.
3. Schützen Sie die Diskette vor magnetischen Feldern.
4. Berühren Sie die Oberfläche der Diskette nicht.
5. Biegen Sie die Diskette nicht.
6. Beschriften Sie die Diskette nicht mit Bleistift oder Füller, sondern benutzen Sie ein vorher beschriftetes, aufklebbares Schildchen.
7. Entfernen Sie eine Diskette nicht aus dem Laufwerk, solange sie sich noch dreht bzw. solange das Lämpchen am Laufwerk noch aufleuchtet.

Das Einlegen einer Diskette in das Laufwerk

Ziehen Sie eine neue Diskette aus ihrer Hülle. Lassen Sie den Diskettenschacht aufspringen, indem Sie die Diskette dagegendrücken. Üben Sie das Öffnen und Schließen des Schachts mehrmals.

Lassen Sie die Diskette in das Laufwerk gleiten, wobei Sie die große ovale Öffnung von sich wegdrehen und den Daumen auf dem Diskettenschildchen halten. Wenn die Diskette ungefähr noch einen Zentimeter hervorschaut, spüren Sie die Ladefeder im Diskettenschacht. Schieben Sie die Diskette mit zwei Fingern weiter, bis es nicht mehr geht. Sie rastet selbsttätig ein. Schließen Sie nun die Tür. Jetzt sollte alles funktionsbereit sein.

Wenn Sie nicht vorsichtig sind, können Sie den Rand Ihrer Diskette beschädigen. Deshalb sollten Sie das Öffnen und Schließen der Tür ohne eingelegte Diskette üben.

Das Formatieren einer neuen Diskette

Wie bereits erwähnt, markieren auf einer Diskette magnetische Signale die Spuren und Sektoren, in denen Information gespeichert wird. Dieser magnetische Markierungsprozeß wird auch als »Formatierung« der Diskette bezeichnet. Eine noch nicht formatierte Platte kann nicht zur Speicherung oder Wiedergewinnung von Informationen verwendet werden, da das Laufwerk nicht weiß, wo die Information gespeichert werden kann.
Legen Sie eine alte oder eine neue Diskette in das Laufwerk, geben Sie

```
OPEN 15,8,15
```

ein und drücken Sie die RETURN-Taste. Tippen Sie dann ein:

```
PRINT#15,"NEW0:<diskettenname>,<id-code>"
```

Als <diskettenname> können Sie einen Ausdruck mit bis zu 8 Zeichen wählen, der <id-code> ist eine aus zwei Zeichen bestehende, beliebige Zahl. Nach Eingabe der letzten Anführungszeichen drücken Sie die RETURN-Taste.
Die soeben beschriebene Prozedur löscht die gesamte Information, die auf der Diskette gespeichert ist; prüfen Sie deshalb, ob Sie die richtige Diskette in der Hand haben.

Laden eines Programms von einer Diskette

Um ein Programm von einer Diskette in den Hauptspeicher Ihres Commodore 64 zu laden, gehen Sie folgendermaßen vor:

1. Schalten Sie den Monitor (Fernsehbildschirm), das Diskettenlaufwerk und den Computer an.

2. Nachdem die rote Betriebsleuchte des Laufwerks erloschen ist, öffnen Sie das Diskettenfach, indem Sie leicht gegen die Klappe drücken.
3. Legen Sie die Diskette vorsichtig mit dem Etikett nach oben und dem länglichen Schlitz nach hinten zeigend ein. Die Diskette muß einrasten.
4. Schließen Sie das Diskettenfach.
5. Tippen Sie LOAD "Programmname",8 in den Computer ein. Der in Anführungszeichen eingeschlossene Programmname sollte der Name des Programms sein, das Sie laden wollen. Drücken Sie die RETURN-Taste.
6. Nachdem das Programm geladen ist (der Computer sendet Ihnen eine READY-Meldung), tippen Sie RUN ein und drücken die RETURN-Taste, um das Programm zu starten.

8. Die Benutzung eines Videomonitors

Wenn Sie anstelle eines Fernsehgeräts einen Videomonitor verwenden, erhalten Sie ein besseres Bild. Wenn Sie wirklich gute Farbgraphik erzeugen wollen, sollten Sie den Commodore Monitor 1701 erwerben.
Der Commodore Videomonitor 1701 besitzt einen 36 cm-Farbbildschirm, der unter Verwendung entweder eines Zwei-Kanal- oder eines Vier-Kanal-Audio/Videokabels direkt mit dem Commodore 64 verbunden werden kann. Bei einem Zwei-Kanal-Kabel wird die Vorderseite des Monitors benutzt.
Stecken Sie das Zwei-Kanal-Kabel in den Audio-Video-Anschluß Ihres Commodore 64, und schalten Sie Computer und Monitor ein. Auf der Rückseite des Monitors befindet sich ein mit FRONT und REAR markierter Schalter. Stellen Sie ihn auf FRONT.
Stecken Sie probehalber jeden der Phonostecker in die an der Vorderseite befindliche VIDEO-Buchse, um herauszufinden, über welchen die Bildsignale laufen. Das andere Kabel übermittelt den Ton. Es läßt sich in die ebenfalls an der Vorderseite befindliche AUDIO-Buchse hineinstecken.
Die Anschlüsse sind identisch mit denen anderer Videomonitore. Falls Ihr Monitor jedoch keinen Lautsprecher besitzt, müssen Sie die Audiokabel bis zu Ihrer Stereoanlage verlängern. Dazu verbinden Sie das Audiokabel mit einem längeren Hi-Fi-Kabel, das mit den üblichen Phonostekkern versehen ist.
Verfügen Sie über ein Vier-Kanal-Audio/Video-Kabel, dann schließen Sie es an der Rückseite des Monitors 1701 an.
Überprüfen Sie, ob der Schalter auf der Rückseite auf REAR steht. Wenn der Commodore 64 und der Monitor eingeschaltet sind, ermitteln Sie, über welchen der vier

Stecker das Helligkeitssignal läuft, indem Sie nacheinander alle vier Stecker in die Phonobuchse auf der Rückseite einstecken, die mit LUMA (für »Luminance Information«) gekennzeichnet ist. Natürlich müssen Sie jedesmal den Bildschirm beobachten, ob sich etwas tut. Zwei der vier Stecker vermitteln Ihnen ein Bild, aber nur das Helligkeitssignal bringt ein normales Schwarzweiß-Bild zustande.

Das andere Kabel, das Sie zur Erzeugung eines Bildes benötigen, übermittelt das Farbsignal. Stecken Sie es in die mit CHROMA gekennzeichnete Buchse; nun sollte Ihr Bildschirm blaue Farbe zeigen. Ist das nicht der Fall, dann vertauschen Sie die beiden Kabel.

Der 1701 verfügt über insgesamt sieben Regler. Belassen Sie die vier, die sich ganz links befinden, in ihrer mittleren Position. Auf diese Weise erhalten Sie das beste Bild.

Der Knopf ganz rechts regelt die Lautstärke. Die anderen beiden Knöpfe auf der rechten Seite dienen zur Steuerung des Bildes in der Waagrechten und in der Senkrechten.

9. Die Benutzung eines Druckers

Sie werden bald einen Drucker an Ihr System anschließen wollen. Commodore stellt mehrere Drucker zur Verfügung: Modell 1515, Modell 1525 und ein neues Modell, MPS 801. Sie entsprechen im wesentlichen dem 1525, der im folgenden beschrieben wird:

Der Commodore Drucker 1525 verfügt über eine Druckzeile von 80 Zeichen und kann 30 Zeichen pro Sekunde drucken. Er erzeugt die Zeichen durch Nadeln, von denen fünf mal sieben auf einer Matrix angeordnet sind; Drucker dieser Art nennt man Punkt-Matrix-Drucker. Wenn Sie die gedruckte Seite genau betrachten, können Sie die einzelnen Punkte sehen.

Ein Vorteil des 1525 besteht darin, daß er alle von C 64 verwendeten Sonderzeichen so druckt, wie sie auf dem Bildschirm erscheinen. Andere Drucker leisten dies nur, wenn die Schnittstelle entsprechend angepaßt wurde.

Das Druckerkabel wird in den seriellen Anschluß eingesteckt, also in die linke der beiden Buchsen in der Mitte der Rückseite des Commodore 64. Der Stecker verfügt über sechs winzige Stifte sowie über eine Abschirmung und ist identisch mit dem Stecker für das Diskettenlaufwerk. Schließen Sie das Druckerkabel an und achten Sie dabei darauf, daß die kleine Kerbe nach oben zeigt.

Schalten Sie nun den Drucker ein. Jetzt leuchtet die rote Kontrollampe auf.

Verwenden Sie für Ihren Drucker nach Möglichkeit Papier mit Randlochung. Nach dem Abschalten des Druckers heben Sie die durchsichtige Abdeckhaube ab und öffnen die Papierführung. Schieben Sie das Papier von hinten in den Papiereinführungsschlitz, bis es vorne am Drucker herausschaut. Richten Sie die Papierführung vorsichtig so ein, daß die Zähne des Papiertransporters in die Perforie-

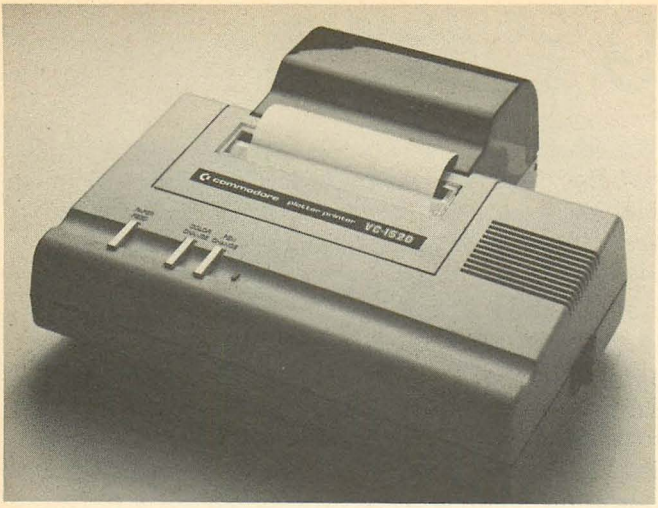


Abb. 4: Der VC 1520 Plotter Printer

rung des Papiers greifen, und schließen Sie die Papierführung wieder. Das Papier sollte nicht zu fest und nicht zu locker sitzen. Ihr gesunder Menschenverstand wird Ihnen bei alledem helfen.
Nun müssen Sie einen Kanal von Ihrem C 64 zum Drucker eröffnen. Es handelt sich dabei um eine Operation des Betriebssystems, mit der Sie nichts zu tun haben. Geben Sie lediglich das Kommando

```
OPEN 1,4
```

ein und drücken Sie die RETURN-Taste. Daraufhin wird alles erledigt.
Die meisten der von Ihnen erworbenen Programme erledigen das für Sie. Wenn Sie aber selbst programmieren, müssen Sie den Kanal auch selbst eröffnen.

Soll die Ausgabe nicht an den Bildschirm, sondern an den Drucker übermittelt werden, geben Sie das Kommando:

```
CMD 1
```

Vergessen Sie aber nicht, die RETURN-Taste zu betätigen. Um die Ausgabe wieder auf den Bildschirm zu bekommen, tippen Sie folgendes Kommando ein und drücken die RETURN-Taste:

```
PRINT#1
```

Wollen Sie den Kanal wieder schließen, dann geschieht das durch dieses Kommando:

```
CLOSE 1
```

Wenn Sie nun ein BASIC-Programm laufen lassen und eine bestimmte Ausgabe an den Drucker schicken wollen, dann benutzen Sie das BASIC-Kommando PRINT#1 anstelle von PRINT, und die Sache funktioniert.
Verwenden Sie einen anderen Drucker als den Commodore angebotenen, dann benötigen Sie eine spezielle Schnittstellenkarte, die die seriellen Daten von Ihrem C64 in parallele Daten für Ihren Drucker umwandelt. Vorteilhaft ist die Kombination einer Cardco-Schnittstelle mit einem Epson-Drucker. Für Schulen empfiehlt sich der Epson-Drucker, da er für eine ständige starke Auslastung vorgesehen ist.
In jedem Fall sollten Sie auch hier darauf achten, daß das von Ihnen erworbene Gerät mit Ihrem Commodore 64 verträglich ist. Drucker werden in zahlreichen Versionen und Preisklassen angeboten und es hängt von Ihrem Verwendungszweck ab, mit welchem Modell Ihnen am besten gedient ist. Die Herstellerangaben über die Arbeitsge-

schwindigkeit der Geräte sind nicht unbedingt zutreffend; in der Regel beziehen sie sich auf Endloszeilen. Der Sprung in eine neue Zeile kostet natürlich Zeit, so daß Drucker in der Regel 10 bis 20 Prozent langsamer sind, als von den Herstellern angegeben. Bidirektionale Drucker, die nach rechts wie nach links schreiben können, besitzen hier einen Vorteil, da sie nicht erst eine ganze Zeile zurückgehen müssen, um weiterschreiben zu können.
Sofern Sie vor allem auf technische Zeichnungen angewiesen sind, benötigen Sie einen Plotter Printer. Er verwendet spezielle Farbstifte zur Erzeugung komplexer Grafiken, wobei er, anders als die übrigen Drucker-Typen, das Blatt nicht nur Zeile für Zeile beschriften kann, sondern sich je nach Bedarf darauf bewegt.

10. Programmieren mit Eingabe

Geben Sie das Kommando NEW ein und drücken Sie die RETURN-Taste. Dadurch wird ein eventuell noch im Hauptspeicher stehendes Programm gelöscht. Dann geben Sie das folgende Programm ein:

```
10 REM NAMENSPROGRAMM
20 PRINT "WIE HEISST DU?"
30 INPUT N$
40 PRINT "DANKE, "?;N$
```

Anschließend tippen Sie LIST ein und drücken wieder die RETURN-Taste. Ihr Programm wird daraufhin auf dem Bildschirm erscheinen. Jetzt geben Sie das Kommando RUN ein und bedienen erneut die RETURN-Taste. Die folgenden Zeilen werden auf Ihrem Bildschirm sichtbar:

```
WIE HEISST DU?
?ALBERTINE
DANKE, ALBERTINE
```

Das Fragezeichen am Anfang der Zeile nach WIE HEISST DU? zeigt an, daß die Maschine auf die Eingabe einer Zeichenreihe wartet. Tippen Sie irgendetwas ein, und drücken Sie die RETURN-Taste.
In Zeile 30 bewirkt das Kommando INPUT N$, daß die Maschine das, was Sie eintippen, im Hauptspeicher ablegt, und zwar von dem Speicherplatz an, der dem Namen N$ entspricht.
Die PRINT-Anweisung in Zeile 40 druckt das, was in Anführungszeichen steht, in genau der Weise, in der Sie es im

Programm geschrieben haben. Eine Variable oder ein Ausdruck, der nicht von Anführungszeichen eingeschlossen ist, wird ausgewertet, und das Ergebnis ausgegeben. Wenn es sich bei dem Ausdruck um einen einfachen Variablennamen handelt, wie bei N$, dann wird der unter N$ abgespeicherte Wert ausgedruckt. Was auch immer Sie also bei der Ausführung von Zeile 30 eingetippt haben, wird nach dem DANKE ausgedruckt.

Übungen

1. Schreiben Sie ein Programm, das Sie nach Ihrer Lieblingsfarbe fragt und dann den folgenden Einzeiler ausdruckt:

```
<farbe>, <farbe>, OH WIE SCHOEN IST <farbe>
```

   Dabei handelt es sich bei <farbe> um die Farbe, die Sie wählen.
   Hier ist ein Beispiel der von dem Programm erzeugten Ausgabe, wenn Sie »LILA« eingetippt haben:

```
RUN
WAS IST DEINE LIEBLINGSFARBE?
?LILA
LILA, LILA, OH WIE SCHOEN IST LILA
```

2. Schreiben Sie ein Programm, das Sie nach Ihrer Lieblingszahl fragt und dann folgendes ausdruckt:

```
TUT MIR LEID, <zahl> WAR NICHT DIE ZAHL, AN DIE ICH
GEDACHT HABE; SIE SCHULDEN DIESEM RECHNER EINE MARK.
```

3. Geben Sie das folgende Programm ein und lassen Sie es laufen:

```
10 INPUT A, A%, A$
20 PRINT A, A%, A$
```

Das Programm erwartet drei Dateneingaben: eine Dezimalzahl, eine ganze Zahl und eine Zeichenreihe. Geben Sie zuerst alle drei Daten auf einmal ein, wobei Sie nach dem ersten und dem zweiten Datum je ein Komma setzen. Wenn Sie dann die RETURN-Taste drücken, werden alle drei Daten gedruckt. Lassen Sie das Programm nun mit Hilfe von RUN noch einmal laufen, geben Sie diesmal nur ein Datum ein, und drücken Sie dann die RETURN-Taste. Das Programm wird weitere Daten von Ihnen verlangen, indem es zwei Fragezeichen (??) ausgibt. Machen Sie weiter, und geben Sie die fehlenden Daten ein. Experimentieren Sie noch etwas, indem Sie falsche Datentypen eingeben. Geben Sie beispielsweise einen Namen ein, wenn das Programm eine Zahl erwartet, und betrachten Sie sich die daraufhin erscheinende Fehlermeldung.

4. Wenn Ihnen Übung 3 noch Schwierigkeiten bereitet, probieren Sie folgendes Programm:

```
10 INPUT A%, B%, C%
```

Geben Sie, wie die Maschine es durch Fragezeichen verlangt, die Zahlen 8, 3 und 7 ein; dies führt zu folgendem Ergebnis:

```
RUN
?8
??3
??7
8      3      7
READY
```

5. Bei der Verwendung von Variablennamen können Sie Überraschungen erleben. Probieren Sie folgende Eingabe:

```
FORDERUNG
```

Sie werden eine Fehlermeldung erhalten, da sich in dem Variablennamen das BASIC-Schlüsselwort FOR verbirgt. Überprüfen Sie durch Eingabe in den Computer, welche der folgenden Variablennamen zulässig ist, und schlagen Sie anschließend in Kapitel 21 nach, welches BASIC-Schlüsselwort für die Fehlermeldung verantwortlich sein kann:

```
TOTAL   FREUND   SORTE   SPENDE   NOTE   SENDER
GETEILT   GLAS   BEITRAG   WAND   WOHNUNG   POST
GEWICHT   WAND   BEZUEGE   FUENF   BUSREISE   CLUB
BUERO   AUSZAHLUNG   VERWENDUNG   FERNSCHREIBER
```

11. Einfache Berechnungen

Ihr Commodore 64 ist sehr gut im Rechnen. Hier ist ein Beispiel:

```
NEW
10 REM KREISFLAECHE
20 PI=3.14159
30 PRINT "WAS IST DER RADIUS IHRES"
40 PRINT "KREISES IN CM"
50 INPUT R
60 FLAECHE=PI*R^2
70 PRINT "DIE FLAECHE IHRES KREISES"
80 PRINT "IST ";FLAECHE;" QUADRAT-CM"
RUN
WAS IST DER RADIUS IHRES
KREISES IN CM
?45
DIE FLAECHE IHRES KREISES
IST 6361.71976 QUADRAT-CM

READY
```

Es folgt eine Beschreibung des Programms Zeile für Zeile, um die bisherigen Schritte zu wiederholen. Wenn Sie das Programm bereits verstanden haben, dann können Sie sofort zu den Übungen am Ende dieses Kapitels übergehen.
Zeile 10 enthält eine REM-Anweisung (»Remark«); sie hat keinerlei Auswirkung auf die Ausführung Ihres Programms.
Zeile 20 stellt eine LET-Anweisung dar, auch arithmetische Zuweisung genannt. In unserem Programmbeispiel nimmt sie die Zahl 3.14159 und speichert sie in dem Speicherplatz ab, der zur Variablen π gehört.

Die Zeilen 30 und 40 sind einfache Druckanweisungen: beide veranlassen, daß jeweils nur eine Zeichenreihe gedruckt wird. Die Anführungszeichen werden nicht ausgegeben.
Zeile 50 zeigt eine INPUT-Anweisung. Da der Variablenname auf einen Buchstaben endet, erwartet das System eine Zahl. Wenn das System diese Anweisung liest, gibt es auf dem Bildschirm ein Fragezeichen am Anfang der nächsten Zeile aus. Dann wartet es und träumt vor sich hin, bis Sie es mit dem Anschlagen der RETURN-Taste wieder aufwecken. Es überprüft dann, ob Sie auch eine echte Zahl eingegeben haben. Wenn das der Fall ist, wird diese Zahl in R abgespeichert. Zeile 60 ist wieder eine arithmetische Zuweisung (LET). Diesmal ist der Ausdruck auf der rechten Seite ein arithmetischer Ausdruck. Er sagt dem System, daß π mit dem Quadrat der Radiuslänge multipliziert werden soll.
Das System weiß auch, daß es die Potenzierung vor der Multiplikation ausführen muß, weil man ihm beigebracht hat, daß Potenzierungsoperationen grundsätzlich zuerst berechnet werden.
Zeile 80 stellt wieder eine Druckanweisung dar, die diesmal aber etwas komplizierter ist. Zuerst wird das Wort IST, gefolgt von einem Leerzeichen, gedruckt. Dann erscheint die Zahl, die den Wert der Variablen FLAECHE ausdrückt. Wenn es sich dabei um eine positive Zahl handelt, ist das nächste Zeichen ein Leerzeichen; geben Sie aber eine negative Zahl (also kleiner als Null) ein, dann wird ein Minuszeichen (—) gedruckt.
Die Zahl endet nicht auf Null. Das Semikolon (;) unmittelbar hinter FLAECHE befieht dem System, mit dem Drukken des nächsten Elements unmittelbar nach der Ausgabe der letzten Ziffer des Werts von FLAECHE zu beginnen.
Als nächste Element erscheint in diesem Fall eine Zeichenreihe, deren erstes Zeichen ein Leerzeichen ist. Sie

sollten nach dem Drucken einer Zahl immer mindestens eine Leerstelle lassen, damit die Zahl nicht mit dem nachfolgenden Druckelement verbunden wird.

Übungen

1. Das Volumen einer Kugel mit Radius R errechnet sich nach der Formel $^3/_4 \cdot \pi \cdot R^3$. Ändern Sie die Zeilen 60 bis 80 des obigen Programms so ab, daß das Programm das Volumen einer Kugel berechnet. Wieviel Kubikzentimeter beträgt das Volumen einer Kugel mit einem Radius von 47 cm?
2. Schreiben Sie eine PRINT-Anweisung, die den Ausdruck 4,235 · 3562,9802 berechnet.
3. Schreiben Sie ein Programm, mit dessen Hilfe Sie den Zinseszins eines fiktiven Betrags berechnen. Angenommen, Sie investieren einen Betrag C bei einer Zinsrate R für einen Zeitraum von N Jahren, so errechnet sich der Zinseszins nach der Formel:
   $S = C(1+R/100)^n$

   Zur Berechnung benötigen Sie folgende Eingabe:

```
100 REM ZINSESZINS
110 PRINT "ANLAGEKAPITAL"
120 INPUT C
130 PRINT "JAEHRLICHE ZINSRATE"
140 INPUT R
150 PRINT "ANLAGEDAUER IN JAHREN"
160 INPUT N
170 S=C*(1+R/100)↑N
180 PRINT "NACH"; N; "JAHREN ERHALTEN
    SIE"; S; "DM"
230 END
```

Bei einem Anlagekapital von DM 10.000,–, einer Anlage-

dauer von 8 Jahren und einer Zinsrate von 17 Prozent erhalten Sie folgende Ausgabe:

```
RUN
ANLAGEKAPITAL
?
10000
JAEHRLICHE ZINSRATE
?
17
ANLAGEDAUER IN JAHREN
?
8
NACH 8 JAHREN ERHALTEN SIE 35114.5 DM
```

12. Arithmetische Ausdrücke

Auf der Tastatur des Commodore 64 werden fünf arithmetische Operationen durch jeweils ein Zeichen dargestellt:

↑ Potenzierung
* Multiplikation
/ Division
\+ Addition
– Subtraktion

Arithmetische Operationen berechnet der Computer in einer bestimmten Reihenfolge, die man auch als »Präzedenzreihenfolge« bezeichnet:
1. Potenzierung
2. Multiplikation und Division
3. Addition und Subtraktion

Wenn Operationen dieselbe Präzedenzstufe haben, wie Multiplikation und Division oder Addition und Subtraktion, dann werden sie von links nach rechts ausgeführt. Man beachte allerdings, daß alle von Klammern eingeschlossenen Operationen zuerst berechnet werden.

Abb. 5: Das Tastenfeld des Commodore 64

In den folgenden Beispielen wird ein in Worten ausgedrücktes Problem in einen arithmetischen Ausdruck umgewandelt. Betrachten Sie die Beispiele, und führen Sie anschließend die Übungen durch.
Beispiel 1: Erheben Sie das Produkt aus 8 und 5 in die dritte Potenz. Antwort: (8*5)↑3.
Beispiel 2: Addieren Sie 2,3,4 und 5, und bilden Sie das Quadrat des Ergebnisses. Antwort: (2+3+4+5)↑2.
Beispiel 3: Ziehen Sie die Wurzel aus $X^2 + Y^2$. Antwort: (X↑2↑,Y↑2) .5 (Wußten Sie schon, daß man die Wurzel aus einer Zahl ziehen kann, indem man sie hoch ½ nimmt?).

Übungen

1. Berechnen Sie den Durchschnitt der Zahlen –4.5, 7.89 und 13.78. Hinweis: Addieren Sie die Zahlen, und teilen Sie die Summe durch drei.
2. Bilden Sie das Produkt aus den Zahlen 1.1, 1.2, 1.3 und 1.4.
3. Für den Anfänger ist es oft mit Schwierigkeiten verbunden, die gewohnte Schreibweise von Formeln in BASIC zu übertragen. Während man normalerweise die Formel

   7ab

   als Produkt »7 mal a mal b« liest, ohne daß das Zeichen für Multiplikation geschrieben wurde, ist dies in BASIC nicht möglich; die korrekte Schreibweise lautet hier:

   7 a b

   Versuchen Sie daher, in folgender Gegenüberstellung selbständig die BASIC-Form des algebraischen Ausdrucks zu schreiben:

Algebra	BASIC
$a + b$	`A + B`
$3(10-3{,}5)$	`3*(10-3.5)`
$\frac{1}{2}(5x+2y)$	`1/2*(5X+2*Y)`
$\frac{a\,b}{c}$	`A*B/C`
$0{,}25(2(x+10)-3y)$	`0.25*(2*(X+10)-3*Y)`
$\frac{a+b}{c+d}$	`A+B/C+D`
x^2+2ab	`X↑2+2*A*B`
x^2+y	`X↑(2+Y)`

Weiterhin existieren in BASIC Schlüsselwörter, die beispielsweise bei der Sinus- oder Wurzelberechnung Anwendung finden; sie sind in Kapitel 21 aufgelistet. Das Schlüsselwort SQR ersetzt so das Wurzelzeichen:

Algebra	BASIC
$\sqrt{x}$	`SQR(X)`
$\sqrt{0{,}5x-y}$	`SQR(0.5*X)-Y`
$\sqrt{\frac{2x+1}{x+y}}$	`SQR(2*X+1)/(X+Y)`

13. IF...THEN-Anweisung

Wollen Sie, daß Ihr Rechner sinnvolle Arbeit für Sie leistet, dann müssen Sie ihn Entscheidungen treffen lassen und ihn auf der Grundlage dieser Entscheidungen antworten lassen. Dafür gibt es die IF...THEN-Anweisung. Sie ist von einfacher Form, wie das folgende Beispiel zeigt.
Tippen Sie das angeführte Programm ein, und lassen Sie es mit Hilfe des Kommandos RUN laufen.

```
NEW
10 REM DER GROESSTE KREIS
20 PRINT "WIE GROSS IST DER GROESSTE KREIS"
30 PRINT "(IN CM), DEN SIE GESEHEN HABEN?"
40 INPUT RADIUS
50 IF RADIUS > 1000 THEN GOTO 80
60 PRINT "SEHR BEEINDRUCKEND"
70 END
80 PRINT "GLAUBE ICH NICHT"
```

In Zeile 50 erscheint eine IF...THEN-Anweisung. Der Ausdruck zwischen dem Wort IF und dem Wort THEN wird »Bedingung« genannt. Sie hat entweder den Wert »wahr« oder den Wert »falsch«. Solche Ausdrücke nennt man »logische Ausdrücke«.
Stößt das Programm auf eine IF-Anweisung, dann wertet es die Bedingung aus. Wenn die Bedingung falsch ist, geht das Programm zur nächsten Zeile über. Ist sie dagegen wahr, führt das Programm das aus, was hinter dem Wort THEN steht.
Geben Sie also einen Radius von weniger oder höchstens 1000 ein, wird das Programm mit SEHR BEEINDRUKKEND antworten und dann anhalten. Wenn Sie einen Radius größer als 1000 eingeben, wird es GLAUBE ICH

NICHT ausgeben und anhalten, weil keine weiteren Anweisungen auszuführen sind.

Übungen

1. Ändern Sie das Programm DER GROESSTE KREIS so ab, daß als Ausgabe GENAU RICHTIG erscheint, wenn Sie die Zahl 1000 eingeben, und dasselbe wie bisher ausgegeben wird, wenn Sie eine Zahl eingeben, die kleiner oder größer als 1000 ist.
2. Schreiben Sie ein Programm, das nach einer Lieblingszahl fragt. Ist die eingegebene Zahl kleiner als 10, soll folgende Ausgabe erscheinen: TUT MIR LEID, SIE IST ZU KLEIN. Auf die Eingabe der Zahl 10 soll es antworten: GENAU RICHTIG. Ist die Zahl größer als 10, soll das Programm ausgeben: SIE IST ZU GROSS. In jedem Fall soll Ihr Programm aufhören, nachdem es eine dieser drei Meldungen ausgegeben hat.

14. Logische Ausdrücke

Jeder logische Ausdruck entspricht einer Frage, auf die es die Antwort »ja« oder die Antwort »nein« gibt.
In logischen Ausdrücken werden folgende Vergleichsmodelle verwendet:

=	gleich
<>	ungleich
<	kleiner als
>	größer als
<=	kleiner als oder gleich
>=	größer als oder gleich

Die Wahrheit oder Falschheit eines logischen Ausdrucks läßt sich dadurch bestimmen, daß man von der Maschine verlangt, den logischen Ausdruck auszugeben. Ist der logische Ausdruck falsch, antwortet die Maschine mit Null; ist er wahr, wird die Maschine die Ausgabe minus 1 (–1) erscheinen lassen. Probieren Sie das mit folgenden Ausdrücken aus:

```
2=2 (wahr)

7<8 (wahr)

45<=23.8 (falsch)

32>14 AND 7=3+4 (wahr)

3*9>100 AND 80>=8*10 (falsch)

50/25=2 AND 400<9*50 (falsch)

45=40+5 OR 6/2<=3 (wahr)

44=7↑2 OR 4↑3>2↑3 (wahr)

11>99/9 OR65=60-5 (falsch)
```

Bei direkter Ausführung kann man Variablen Werte zuweisen. Diese bleiben erhalten, bis Sie sie ändern. Geben Sie die folgende Zeile ein, und drücken Sie dann die RETURN-Taste:

```
X=1: Y=3: Z=5
```

Und nun versuchen Sie, die folgenden drei Ausdrücke zu erhalten:

```
?X*(Y+Z)<(4.56-Z)*(Y-X)
?(X-Y)↑Z<>342 OR X↑X↑X↑X=1
?NOT(2>3ANDNOT(4+2<>6))
```

Bestimmen Sie, damit es mehr Spaß macht, den Wahrheitswert der obigen drei Ausdrücke zuerst selbst, ehe Sie die Maschine rechnen lassen.

Übungen

1. Schreiben Sie einen logischen Ausdruck, der besagt, daß X + 1 nicht gleich Y ist.
2. Schreiben Sie einen logischen Ausdruck, der besagt, daß Y^2 größer als 5 ist, und daß X − 5 kleiner oder gleich Z ist.

15. Schleifen

Ein weiteres wertvolles Paar von Anweisungen sind FOR und NEXT. Sie werden dazu benutzt, Schleifen in Ihrem Programm zu bilden, so daß dieselbe Operation immer wieder ausgeführt werden kann. Für die meisten Menschen sind Wiederholungen langweilig, während Maschinen im Gegensatz dazu sie geradezu lieben.
Erstellen Sie einfach eine Liste der Anweisungen, die Sie wiederholt von Ihrem Computer ausgeführt haben möchten. Setzen Sie eine FOR-Anweisung davor und eine NEXT-Anweisung dahinter, womit Sie der Maschine sagen, wo sie anfangen und wo sie aufhören soll, und lassen Sie dieses Programmstück laufen.
Versuchen Sie einmal dieses Programm:

```
NEW
 5 REM ZAEHLEN BIS 10
10 PRINT "ICH ZAEHLE BIS 10 IN ZWEIERSCHRITTEN"
20 FOR COUNT=2 TO 10 STEP 2
30 PRINT COUNT
40 NEXT COUNT
RUN
ICH ZAEHLE BIS 10 IN ZWEIERSCHRITTEN
2
4
6
8
10

READY
```

Die Variable COUNT in Zeile 20 wird »Schleifenzähler« genannt. Es gibt nur einen Schleifenzähler pro Schleife,

und man kann seinen Wert innerhalb der Schleife nicht verändern. Man kann den Wert des Schleifenzählers in Berechnungen verwenden, aber er darf nie auf der linken Seite einer Zuweisung stehen.
In der FOR-Anweisung müssen Sie angeben, mit welchem Wert der Schleifenzähler anfangen und mit welchem Wert er aufhören soll. Wenn Sie sonst nichts angeben, wird bei jedem Durchlauf der Schleife die Zahl 1 zu dem im Schleifenzähler angegebenen Wert hinzugezählt. Wenn Sie wollen, daß ein anderer Wert jedes Mal dazuaddiert wird, dann müssen Sie noch das Wort STEP, gefolgt von der gewünschten positiven oder negativen Zahl, hinzufügen.
Zeile 30 ist die einzige PRINT-Anweisung innerhalb der Schleife. In diesem Fall druckt das System den Wert des Schleifenzählers, den er jeweils beim Durchlaufen dieser Anweisung hat.
Alle Schleifen müssen mit einer NEXT-Anweisung enden. Es ist guter Stil, aber nicht notwendig, hier noch einmal den Schleifenzähler, der erhöht und getestet werden soll, anzugeben. Das zahlt sich vor allen Dingen für denjenigen aus, der das Programm liest.
Wenn das System auf eine NEXT-Anweisung läuft, addiert es zunächst die hinter STEP angegebene Zahl auf den Schleifenzähler. Dann entscheidet es, ob es noch einmal durch die Schleife läuft oder ob es damit aufhört.
Wenn die hinter STEP angegebene Schrittweise positiv ist, und wenn der erhöhte Wert des Schleifenzählers die obere Grenze noch nicht überschreitet, dann wird mit der Anweisung, die auf die FOR-Anweisung folgt, fortgefahren. Wenn jedoch die obere Grenze überschritten worden ist, dann wird mit der Anweisung unmittelbar hinter der NEXT-Anweisung fortgefahren.
In einer Schleife kann eine weitere Schleife stehen. In diesem Fall spricht man davon, daß die Schleifen ineinander geschachtelt sind. In diesem Fall werden jedesmal, wenn

die äußere Schleife einmal durchlaufen wird, sämtliche Schleifendurchläufe der inneren Schleife ausgeführt. Ineinander geschachtelte FOR...NEXT-Schleifen sind immer dann nützlich, wenn eine Menge von Aktionen als Bestandteil anderer sich wiederholender Aktionen mehrfach ausgeführt werden muß.
Häufig wird der Schleifenzähler in Berechnungen innerhalb von FOR...NEXT-Schleifen verwendet. Das ist zulässig, solange der Wert des Schleifenzählers innerhalb der Schleife nicht verändert wird.
Geben Sie das folgende Programm, das die Multiplikationstabelle aller Zahlen zwischen 1 und 12 ausdruckt, ein, und lassen Sie es mit Hilfe von RUN laufen. Denken Sie daran, daß immer eine Schrittweite von 1 angenommen wird, wenn STEP nicht angegeben ist.

```
NEW
10 REM MULTIPLIKATIONSTABELLE
20 FOR X=1 TO 12
30 PRINT X;"-TABELLE":PRINT
40 FOR Y=1 TO 12
50 PRINT X;" MAL ";Y;" IST ";X*Y
60 NEXT Y
70 PRINT
80 NEXT X
```

Während der Wert von Y die Zahlen von 1 bis 12 annimmt, bleibt der Wert von X konstant. Anschließend wird der Wert von X um 1 erhöht, und der Schleifendurchlauf wiederholt sich.
Es ist vorteilhaft, die Ausgabe dieses Programms an den Drucker zu schicken, denn auf dem Bildschirm wird sie zu schnell angezeigt, als daß man sie wirklich verfolgen könnte. Die Ausführung des Programms kann jedoch verlangsamt werden, indem Sie eine spezielle Form der FOR...NEXT-Schleife verwenden, die man »Verzöge-

rungsschleife« nennt. Fügen Sie dem Programm die folgenden beiden Zeilen hinzu:

```
65 FOR DELAY=1 TO 500
67 NEXT DELAY
```

Diese beiden Zeilen haben zur Folge, daß Ihr Programm solange anhält, wie der Computer braucht, um bis 500 zu zählen. Die Dauer der Verzögerung kann dadurch gesteuert werden, daß man den Endwert erhöht oder erniedrigt.

Übungen

1. Schreiben Sie ein Programm, das hundertmal JONATHAN druckt.
2. Berechnen Sie die Summe der Zahlen 1,2,3,4,...,100; d.h., zählen Sie die ersten hundert ganzen Zahlen zusammen. Hinweis: Verwenden Sie eine Schleife mit einem Zähler, der von 1 bis 100 läuft, und zählen Sie bei jedem Schleifendurchlauf den Wert des Zählers zu dem, was Sie bisher schon zusammengezählt haben, dazu, wobei Sie eine Anweisung wie die folgende verwenden:

```
SUMME = SUMME + ZAEHLER
```

3. Bestimmen Sie die Summe der Zahlen 5,10,15,...190, 195,200. Hinweis: Verwenden Sie dieselbe Technik wie in Übung 2, aber geben Sie eine Schrittweite von 5 ein.
4. Schreiben Sie ein Programm, das Additionstabellen für die Zahlen von 0 bis 9 ausdruckt.
5. Schreiben Sie ein Programm, das die Multiplikationstabelle der Zahl 7 erbringt, jedoch immer nur Vielfache

gerader Zahlen von 7 anzeigt. Antwort: Ändern Sie Zeile 40 des auf Seite 58 abgedruckten Programms wie folgt:

```
40 FOR Y=2 TO 12 STEP 2
```

Wie müßte die Veränderung des Programms aussehen, um nur Vielfache ungerader Zahlen von 7 zu erhalten?

6. Schreiben Sie ein Programm, das rückwärts zählt, beispielsweise von 20 auf 0. Ein Hinweis: Sie benötigen dazu eine negative Schrittgröße, in dem gewählten Beispiel als STEP –1, um alle Zahlen von 20 bis 0 erfassen zu können. Welche Schrittgröße müssen Sie wählen, um ausschließlich alle geraden Zahlen dabei zu erhalten?

16. Speicherung von Daten in Programmen

Manchmal ist es nützlich, viele Daten durch ein Programm bearbeiten zu lassen. Auf diese Weise kann man Programm und Daten zugleich sichern.
DATA-Anweisungen gestatten die Speicherung großer Mengen von Information innerhalb des Programms. DATA-Anweisungen können ganze Zahlen, Dezimalzahlen oder Zeichenreihen enthalten. Sie werden dazu verwendet, den in READ-Anweisungen enthaltenen Variablen Werte zuzuweisen, z.B. auf folgende Weise:

```
10 READ X,Y%,Z$
20 DATA 25.4,6,"JAHR"
30 PRINT X,Y%,Z$
RUN
25.4      6       JAHR
```

Wie in obigem Beispiel muß der Typ der Elemente in der DATA-Anweisung mit dem Typ der Variablen, denen die Werte zugewiesen werden sollen, übereinstimmen. So wäre es unzulässig, die Dezimalzahl 25.4 der Zeichenreihenvariablen Z$ zuzuweisen.
Es muß keine eindeutige Korrespondenz zwischen den DATA-Anweisungen und den READ-Anweisungen bestehen. Die Elemente in den DATA-Anweisungen werden behandelt, als würden sie eine einzige lange Liste darstellen, entsprechend der aufsteigenden Reihenfolge ihrer Zeilennummern. Sobald ein Element von dieser Liste gelesen worden ist, wird es quasi davon gestrichen. Die folgenden drei Zeilen sind äquivalent zu den obigen und ebenso zulässig:

```
10 READ X
20 READ Y%,Z$
100 DATA 25.4,6,"JAHR"
```

Auch folgendes Programm ist möglich:

```
10 READ X,Y%,Z$
100 DATA 25.4
110 DATA 6
120 DATA "JAHR"
```

Jetzt versuchen Sie einmal dieses Beispiel:

```
NEW
10 ?"WARE","ANZ","PREIS","KOSTEN"
20 ?
30 READ WARE,ANZ%,PREIS
40 KOSTEN=ANZ%*PREIS
50 ?WARE$,ANZ%,PREIS,KOSTEN
140 DATA PAPIER,5
150 DATA 1.25,BROT,12,3.99,TISCH
```

Beachten Sie, daß eine READ-Anweisung zwei DATA-Anweisungen verwendet. Außerdem gibt es überzählige DATA-Elemente. Beides ist völlig zulässig.
Stehen dem Programm keine Datenwerte mehr zur Verfügung, bevor sämtliche READ-Variablen mit Werten besetzt sind, bekommt man eine Fehlermeldung.
Vielfach werden die Elemente einer DATA-Anweisung von einer READ-Anweisung, die innerhalb einer Schleife liegt, gelesen. Fügen Sie dem obigen Programm die folgenden Zeilen hinzu:

```
25 FOR K=1 TO 4
60 NEXT K
160 DATA 3,50.55,STUHL,5,25.15
```

Die FOR...NEXT-Schleife in obigem Programm ist nur für Fälle geeignet, in denen genau vier Datenmengen gelesen werden müssen. Vielfach soll jedoch ein und dasselbe Programm mit verschieden vielen Daten laufen. In solchen Fällen wird die letzte Datenmenge dazu verwendet, das Ende der Daten anzuzeigen. Eine solche Anzeige kann ein beliebiger Wert sein, den gültige Daten niemals annehmen würden. Nach jeder READ-Anweisung überprüft das Programm, ob das Ende der Daten angezeigt worden ist, ehe es die weiteren Operationen innerhalb der Schleife ausführt. Bei Anzeige des Datenendes wird die Schleife beendet.
Nehmen Sie jetzt Änderungen an dem letzten Programm vor: Löschen Sie die Zeilen 25 und 60, indem Sie jede der beiden Zeilennummern, gefolgt vom Anschlagen der RETURN-Taste, eingeben. Dann fügen Sie die folgenden Zeilen hinzu:

```
50 IF PREIS>0 THEN ? WARE$,ANZ%,PREIS
CST:GOTO 30
170 DATA 0,0,0
```

Beachten Sie, daß für alle drei einzulesenden Daten Werte angegeben sind. Wenn die READ-Anweisung nämlich nicht für alle Variablen Werte findet, dann bricht der Computer das Programm mit einer Fehlermeldung ab.

17. Ein Spesenabrechnungsprogramm

Sie sind jetzt in der Lage, ein Programm zu schreiben, das Sie zu nützlichen Berechnungen verwenden können. Bei folgender Problemstellung werden zehn DATA-Anweisungen verwendet. Jede Anweisung enthält den Typ T, den Betrag A und den Rechnungssteller V$ für die Abrechnung einer Dienstreise nach München.

Das Programm soll folgendes leisten:

1. Der jeweilige Erstattungsbetrag R berechnet sich wie folgt:
   Wenn T gleich 1 ist, dann ist R 100 % des Betrags von A.
   Wenn T gleich 2 ist, dann ist R 80 % des Betrags von A.
   Wenn T gleich 3 ist, dann ist R 20 % des Betrags von A.
   Wenn T irgendeinen anderen Wert hat, dann soll die Ausgabe FALSCHE TYPZAHL gedruckt werden.
2. Es soll der Typ T, der Rechnungssteller V$, der Betrag A und der Erstattungsbetrag R (der in Schritt 1 berechnet worden ist) ausgegeben werden.
3. Sind alle zehn Anweisungen verarbeitet worden, soll das Programm VERARBEITUNG ABGESCHLOSSEN ausgeben und anhalten.

Das Programm sieht folgendermaßen aus:

```
100 REM SPESENABRECHNUNG
110 REM ES FOLGEN 10 DATA-ANWEISUNGEN, JEDE ENTHAELT
120 REM TYP, BETRAG, RECHNUNGSSTELLER
130 DATA 1,180.00, "HILTON MUENCHEN"
140 DATA 3,21.50, "CITY BAR"
150 DATA 2,32.80, "PIZZALAND"
160 DATA 1,12.60, "CAFE"
170 DATA 1,8.70, "TAXI"
180 DATA 3,10.00, "STADTRUNDFAHRT"
190 DATA 1,180.00, "HILTON MUENCHEN"
200 DATA 2,8.20, "ZEITSCHRIFTEN"
```

```
210 DATA 2,9.00, "TAXI"
220 DATA 3,55.40, "CITY BAR"
230 REM *****ENDE DER DATEN*****
```

Sichern Sie diesen Teil des Programms, bevor Sie weitermachen. Wir werden ihn nämlich in späteren Programmen noch benötigen. Setzen Sie dann das Programm folgendermaßen fort:

```
240 PRINT "SPESENABRECHNUNG"
250 PRINT "T";TAB(2);"RECHNUNGSSTELLER";
TAB(26);"BETRAG";TAB(34);"ERST"
260 FOR I=1 TO 10
270 READ T,A,V$
280 IF T=1 THEN R=A
290 IF T=2 THEN R=.80*A
300 IF T=3 THEN R=.2*A
310 IF T>3 OR T<1 THEN PRINT "FALSCHE
TYPZAHL" :R=0
320 PRINT T;TAB(2);V$;TAB(26);A;TAB
(34);R
330 NEXT I
340 PRINT "VERARBEITUNG ABGESCHLOSSEN"
350 END
```

Testen Sie das Programm, und lassen Sie es laufen. Sie sollten folgende Ausgabe erhalten:

```
RUN
SPESENABRECHNUNG
T  RECHNUNGSSTELLER          BETRAG          ERST
1  HILTON MUENCHEN           180.00        180.00
3  CITY BAR                   21.50          4.30
2  PIZZALAND                  32.80         26.24
1  CAFE                       12.60         12.60
1  TAXI                        8.70          8.70
3  STADTRUNDFAHRT             10.00          2.00
1  HILTON MUENCHEN           180.00        180.00
```

```
2  ZEITSCHRIFTEN              8.20          6.56
2  TAXI                       9.00          9.00
3  CITY BAR                  55.40         11.08
VERARBEITUNG ABGESCHLOSSEN
```

Wenn Ihr Programm läuft und Sie es verstehen, können Sie zum nächsten Kapitel übergehen. Andernfalls lesen Sie bitte weiter.

Das TAB-Kommando in Zeile 250 ist für Sie vielleicht noch neu. Es bewirkt, daß bei der Ausgabe in die entsprechende Spalte (zwischen 0 und 255) gegangen wird. Das ganz links stehende Zeichen einer Zeile befindet sich immer in Spalte 0; insofern bewirkt TAB(2), das in die dritte Spalte von links gegangen wird.

Nur jeweils eine der vier IF-Anweisungen kann für einen Wert von T wahr sein. Diese IF-Anweisung bestimmt jeweils den neuen Wert von R. Zeile 310 ist ein Beispiel dafür, daß auch komplexere Anweisungen in Abhängigkeit von einer erfüllten Bedingung erleichtert werden können. In diesem Fall, wenn also die Typzahl größer als 3 oder kleiner als 1 ist, druckt das System eine entsprechende Meldung aus und setzt dann den Erstattungsbetrag R auf Null, ehe die entsprechende Zeile ausgegeben wird.

18. Felder

Felder erlauben eine wesentlich schnellere Berechnung der Spesendaten aus München, da man sie nur einmal zu lesen braucht.
Fügen Sie Ihrem Programm die folgenden Zeilen hinzu:

```
240 ?"VERWENDUNG VON FELDERN"
250 REM DIMENSION DER FELDER
260 DIM T(10), A(10), V$(10)
270 REM EINLESEN DER DATEN IN DIE FELDER:
280 FOR P=1 TO 10
290 READ T(P), A(P), V$(P)
300 NEXT P
310 STOP
```

Lassen Sie dieses Programm mit Hilfe von RUN laufen; Sie sollten folgende Ausgabe erhalten:

```
RUN
BREAK IN 310
READY
```

Was bewirkt dieser Teil des Programms? In Zeile 260 werden die Dimensionen von drei Feldern, T, A und V$, festgelegt. Jedes Feld hat zehn Elemente; die Anzahl der Elemente ist die Zahl in den Klammern. Auf diese Weise beauftragen Sie das System, von vornherein Platz für die Felder bereitzustellen.
Zeile 290 weist das System an, das nächste verfügbare Datenelement in das Element P des Feldes T einzulesen, das darauffolgende Datenelement in das Element P des Feldes A und das dann folgende Datenelement in V$(P).
Wenn das Programm läuft, fügen Sie die folgenden Zeilen an:

```
310 REM BERECHNUNG DES DURCHSCHNITTS AVE
320 AVE=0
330 FOR K=1 TO 10
340 AVE=AVE + A(K)
350 NEXT K
360 ?"DURCHSCHNITTSBETRAG IST ",AVE/10
RUN
```

Sie werden jetzt folgende Antwortzeile bekommen:

```
DURCHSCHNITTSBETRAG IST 58.76564
```

Der Durchschnitt wird mit Hilfe der Variablen AVE berechnet, die die Summe der zehn Beträge aus dem Feld A aufnimmt. In Zeile 320 wird AVE auf Null gesetzt und in Zeile 340 wird der Betrag A(K) jeweils zum bisherigen Wert von AVE dazuaddiert, wodurch ein neuer Wert von AVE entsteht. In Zeile 360 sind alle zehn Werte zusammengezählt, so daß sie nur noch durch zehn geteilt werden müssen, woraus sich der korrekte Durchschnittswert ergibt.
Wenn Sie zusätzlich die Namen der Rechnungssteller sehen wollen, erreichen Sie das durch das folgende Programmstück:

```
370 FOR M=1 TO 10
380 ?V$(M)
390 NEXT M
RUN
```

19. Die Formate von Schlüsselwörtern

Mittlerweile existiert eine Sprache, die einen Programmierer genau informiert, welche Anweisungen für ein bestimmtes System zulässig sind. Diese Sprache hat keinen Namen, aber sie wird heutzutage überall verwendet. Sie ist sowohl für den Anfänger als auch für den fortgeschrittenen Programmierer geeignet und wird, wenn Sie weiterhin programmieren wollen, Ihnen eine große Hilfe sein. (Wenn Sie für den Rest Ihres Lebens vom Programmieren genug haben, dann hören Sie am besten hier mit dem Lesen auf und verkaufen dieses Buch an einen Freund.)
Betrachten wir ein so einfaches Schlüsselwort wie NEW. Es bietet keine Wahlmöglichkeiten. Das allgemeine Format dieses Kommandos sieht daher folgendermaßen aus:

```
FORMAT: NEW
```

Anders liegt der Fall bei einem etwas komplizierteren Schlüsselwort wie RUN. In diesem Fall können Sie entweder RUN eingeben und die RETURN-Taste drücken (was Sie bisher immer gemacht haben) oder Sie lassen eine Zeilennummer folgen. Die Ausführung des Programms beginnt dann mit der angegebenen Zeilennummer. Das Format der RUN-Anweisung stellt sich daher folgendermaßen dar:

```
FORMAT: RUN[<zeilennummer>]
```

Die Angabe wirkt auf den ersten Blick etwas irritierend, aber sie ist sehr logisch aufgebaut. Die eckigen Klammern ([...]) zeigen an, daß der von ihnen eingeschlossene Teil

wahlfrei (optional) ist, d.h., Sie können ihn weglassen, wenn Sie wollen. Die spitzen Klammern stehen für veränderbare Ausdrücke, in diesem Fall für Zeilennummern, bei der mit der Ausführung des Programms begonnen werden soll. Was innerhalb der spitzen Klammern steht, soll Ihnen lediglich andeuten, worum es an dieser Stelle geht. Wenn Sie die Liste der Schlüsselwörter durchsehen, dann werden Sie auf Ausdrücke wie <variable>, <zahl>, <speicherplatz>, <dateinummer>, <ausdruck> usw. stoßen. Zwar handelt es sich sowohl bei <zahl> als auch bei <dateinummer> um Zahlen, aber eine <dateinummer> kann den Wert 15 nicht übersteigen.
Noch komplexer wirkt folgendes Format:

```
FORMAT: PRINT [<element>] [,/;<element>]...
```

Hierbei ist <element> ein Wort oder ein Ausdruck, beispielsweise »Spesenberechnung« oder 3+2.
Wie die beiden eckigen Klammern verraten, sind beide Objekte hinter dem Kommando PRINT optional. Daraus folgt zugleich, daß PRINT für sich alleine vom System akzeptiert wird. Es sagt nichts darüber aus, was das System machen wird (tatsächlich wird es eine leere Zeile ausgeben), aber es signalisiert, daß die Syntax in Ordnung ist oder, was dasselbe ist, daß das System weiß, was Sie meinen. Sie wissen vielleicht nicht, was Sie meinen, aber das System weiß es.
In dem ersten eckigen Klammernpaar steht das Objekt <element>. Das bedeutet, daß man ein <element> unmittelbar hinter dem Wort PRINT schreiben kann, etwas anderes aber ausgeschlossen ist. Damit ist also eine Anweisung wie PRINT A$ zulässig, wohingegen PRINT;A$ unzulässig ist.
Das zweite eckige Klammernpaar ist komplizierter und auch schwieriger zu lesen. Der Schrägstrich (/) bedeutet, daß an dieser Stelle die Wahl zwischen den angegebenen

Zeichen oder Zeichenreihen besteht. Die Formatanweisung besagt hier, daß man nach dem ersten Element ein Komma oder ein Semikolon setzen kann.
Was Sie nun auch setzen (ein Komma oder ein Semikolon), ihm muß ein weiteres <element> folgen, denn wenn ein Bestandteil der in den eckigen Klammern aufgeführten Angaben geschrieben wird, dann müssen auch die übrigen angegeben werden. Während also PRINT A,B oder PRINT A;B zulässig ist, erhalten Sie für PRINT A;;;B, eine Fehlermeldung.
Jetzt sollten Sie keine Schwierigkeiten mehr haben, die Syntax eines beliebigen Schlüsselworts zu verstehen.

Regeln für Formate:

1. Alles, was innerhalb eckiger Klammern steht, ist optional.
2. Alles, was innerhalb spitzer Klammern steht, stellt eine Zeichenreihe dar, die veränderbar ist. Innerhalb der spitzen Klammern steht ein aussagekräftiges Wort für diese Zeichenreihe.
3. Jedes Zeichen, das nicht innerhalb spitzer Klammern steht, muß genauso geschrieben werden, wie es erscheint, abgesehen von eckigen oder spitzen Klammern und von Zeichenreihen, die durch Schrägstrich (/) voneinander getrennt sind.
4. Werden durch Schrägstrich (/) zwei oder mehr Zeichenreihen voneinander getrennt, bedeutet dies, daß Sie eine Wahlmöglichkeit zwischen den verschiedenen Zeichenreihen haben.
5. Drei Punkte hinter einem Paar eckiger Klammern geben an, daß Sie das, was innerhalb der eckigen Klammern steht, beliebig oft wiederholen können.

20. Die verschiedenen Arten von Schlüsselwörtern

Ein Schlüsselwort ist eine Zeichenreihe; das System ist so programmiert, daß es Schlüsselwörter sofort erkennt. Beispiele für Schlüsselwörter sind RUN, NEW, LIST, PRINT, OR, COS. Immer wenn das System eine solche Zeichenreihe liest, wird es dadurch veranlaßt, eine bestimmte Operation durchzuführen. Das ist der Grund, warum Sie keine Variablennamen verwenden dürfen, die Schlüsselwörter enthalten: Das System würde grundlos aufgeschreckt werden.
Geben Sie z.B. eine so unschuldig ausssehende Anweisung wie

```
50 COST = QUANTITAET * PREIS
```

ein, wird das System Sie daran erinnern, daß ein Syntaxfehler vorliegt. Der Grund: Sie haben das Wort COS (für die Cosinusfunktion) als die ersten drei Buchstaben Ihres Variablennamen COST verwendet.
Das System muß deshalb so engstirnig sein, da es Ihnen erlaubt ist, Leerzeichen zu schreiben, wo immer Sie wollen. Sie können aber Ihre Anweisungen auch ohne Leerzeichen schreiben. Das System muß deshalb gewisse Zeichenreihen erkennen, auch dann, wenn sie mit anderen Zeichenreihen zusammenstoßen. Diese gewissen Worte sind genau die Schlüsselwörter von Commodore 64 BASIC.
Es gibt verschiedene Arten von Schlüsselwörtern. Zum einen existieren Wörter, die nicht wirklich zur Sprache BASIC gehören; ihnen kann man keine Zeilennummern voranstellen. Dies betrifft Schlüsselwörter wie NEW, RUN und LIST. Sie werden zur näheren Kennzeichnung als Kommandos klassifiziert. Es ist sicher vernünftig, Anwei-

sungen, die Bestandteil eines Programms sein können (im weiteren als »Anweisungen« bezeichnet), zu trennen von Anweisungen, die nicht Bestandteil eines Programms sein können (im weiteren »Kommandos« genannt).
Weiterhin werden Anweisungen (genauer gesagt, die in einem Programm zulässigen Anweisungen) unterschieden in solche, die zusätzliche Geräte, wie Laufwerke oder Drucker, voraussetzen, und solche, die Sie auch dann verwenden können, wenn Sie nichts als einen Schwarzweißfernseher und Ihren Commodore 64 besitzen. Die Anweisungen, die zusätzliche Geldausgaben von Ihnen fordern, nennt man E/A-Anweisungen (»Ein/Ausgabe-Anweisungen«). Mit ihrer Hilfe wird Information an ein Laufwerk, einen Rekorder, einen Drucker oder ein anderes Gerät geschickt oder von dort geholt.
Eine weitere Klasse von BASIC-Anweisungen stellen die Funktionen dar. Wenn Sie das Programmieren wirklich lernen wollen, dann müssen Sie wissen, was eine Funktion ist.
Eine Funktion ist eine Vorschrift, die einem oder mehreren Eingabewerten genau einen Ausgabewert zuordnet. Handelt es sich bei dem Ausgabewert immer um eine ganze Zahl, dann spricht man von einer ganzzahligen Funktion; ist der Ausgabewert immer ein String (also eine Zeichenreihe), dann spricht man von einer Stringfunktion; stellt schließlich der Ausgabewert eine Dezimalzahl dar, bezeichnet man die Funktion als Dezimalfunktion.

Im folgenden sind die verschiedenen Arten von Schlüsselwörtern nochmals zusammengefaßt:

1. Kommando: Systemkommandos wie RUN, LIST oder NEW, die nicht Bestandteil eines Programms sein können.
2. Anweisung: Einfache Programmanweisungen wie LET, PRINT, READ, DATA usw., die keine zusätzlichen Geräte erfordern.

3. Ganzzahlige Funktion: Eine Funktion, die ein ganzzahliges Ergebnis liefert.
4. Dezimalfunktion: Eine Funktion, die eine Dezimalzahl als Ergebnis liefert.
5. Stringfunktion: Eine Funktion, die einen String (Zeichenreihe) als Ergebnis liefert.
6. Operator: Ein logischer Operator, der Bestandteil einer umfassenden Anweisung ist.
7. E/A-Anweisung: Ein Kommando oder eine Anweisung, durch die ein Eingabegerät oder ein Ausgabegerät gesteuert wird.

21. Liste der Schlüsselwörter

ABS
Typ: Dezimalfunktion
Beispiel: PRINT ABS(–4.5)
4.5
Format: ABS(<ausdruck>)
Leistung: Liefert den Absolutbetrag einer Zahl. Bei einer positiven Zahl ist der Absolutbetrag die Zahl selbst, bei einer negativen Zahl die mit –1 multiplizierte Zahl).

AND
Typ: Operator
Beispiel: IF X <3 AND Y>2
Format: <ausdruck> AND <ausdruck>
Leistung: Liefert den Wahrheitswert »wahr« nur dann, wenn beide Ausdrücke wahr sind; ist nur ein Ausdruck wahr oder sind beide falsch, ist der Wahrheitswert der Verknüpfung »falsch«. Wenn Zahlen anstelle der Ausdrücke eingesetzt werden, ist der Funktionswert eine durch 16 Bit darstellbare Zahl, die dadurch entsteht, daß man die beiden Zahlen bitweise durch logisches »und« miteinander verknüpft, nachdem man die beiden Zahlen in eine 16-Bit-Darstellung umgewandelt hat.
Fehlermeldungen: ?ILLEGAL QUANTITY, falls Zahlen verwendet werden, die außerhalb des Bereichs zwischen –32768 und +32767 liegen.

ASC
Typ: Ganzzahlige Funktion
Beispiel: PRINT ASC(»MAX«)
Format: ASC(<string>)
Leistung: Liefert den ASCII-Wert des ersten Zeichens der Zeichenreihe.

Fehlermeldungen: ?ILLEGAL QUANTITY, falls der String leer ist.

ATN
Typ: Dezimalfunktion
Beispiel: ATN(56.0987)
Format: ATN(<zahl>)
Leistung: Berechnet den Arcustangens der angegebenen Zahl. Weitere Winkelfunktionen berechnen COS, SIN und TAN.

CHR$
Typ: Stringfunktion
Beispiel: CHR$(65)
Format: CHR$(<zahl>)
Leistung: Ersetzt Steuerzeichen durch Zahlen im ASCII-Code.
Fehlermeldungen: ?ILLEGAL QUANTITY, falls eine Zahl verwendet wird, die außerhalb des Bereichs zwischen 0 und 155 liegt.

CLOSE
Typ: E/A-Anweisung
Beispiel: CLOSE 4
Format: CLOSE <dateinummer>
Leistung: Speichert eine unvollständige Datei auf ein Gerät, beendet die Kommunikation und gibt den von einem eröffneten Kanal beanspruchten Teil des Hauptspeichers wieder frei.

CLR
Typ: Anweisung
Beispiel: CLR
Format: CLR
Leistung: Löscht alle nicht dauerhaften Daten im Hauptspeicher.

CMD
Typ: E/A-Anweisung
Beispiel: CMD 4, »STEUERBETRAG«
Format: CMD <dateinummer>, [<string>]
Leistung: Lenkt die Normalausgabe vom Bildschirm auf das spezifizierte Gerät bzw. auf die spezifizierte Datei. Ein angegebener String wird als Dateiname aufgefaßt. Die Dateinummer muß in einer vorausgegangenen OPEN-Anweisung bereits vorgekommen sein.
Um die Ausgabe wieder auf den Bildschirm zu lenken, muß mit einer PRINT#-Anweisung eine Leerzeile an das Gerät bzw. die Datei geschickt werden, ehe sie mit CLOSE geschlossen wird. Auch bei Auftreten eines beliebigen Systemfehlers wird die Ausgabe wieder auf den Bildschirm gegeben.

CONT
Typ: Kommando
Beispiel: CONT
Format: CONT
Leistung: Setzt die Ausführung eines Programms an der Stelle fort, an der sie unterbrochen wurde.
Fehlermeldungen: CAN'T CONTINUE, falls das System die Ausführung nicht fortsetzen kann (das kann verschiedene Gründe haben).

COS
Typ: Dezimalfunktion
Beispiel: COS(Y*PI/180)
Format: COS(<zahl>)
Leistung: Berechnet den Cosinus einer Zahl.

DATA
Typ: Anweisung
Beispiel: DATA »WOZU«, 865, 67.35
Format: DATA <konstantenliste>

Leistung: Speichert die aufgezählten Konstanten innerhalb des Programms.

DEF FN
Typ: Anweisung
Beispiel: DEF FN A(X)=INT(RND(1)*X+1)
Format: DEF FN <name>(<variable>)= <ausdruck>
Leistung: Vereinbarung einer vom Benutzer definierten Funktion.

DIM
Typ: Anweisung
Beispiel: DIM A(30), B(25)
Format: DIM <variable>(<indices>)
[,<variable>(<indices>)]...
Leistung: Reserviert den erforderlichen Hauptspeicher für ein oder mehrere Felder.
Fehlermeldungen: REDIM'D ARRAY, falls versucht wird, ein Feld innerhalb eines Programms zum zweiten Mal zu dimensionieren.

END
Typ: Anweisung
Beispiel: END
Format: END
Leistung: Beendet die Ausführung eines Programms und bewirkt die Ausgabe von READY auf dem Bildschirm.

EXP
Typ: Dezimalfunktion
Beispiel: EXP(1)
Format: EXP(<zahl>)
Leistung: Berechnet die mathematische Konstante e (e=2.71828183) hoch die angegebene Zahl.
Fehlermeldungen: ?OVERFLOW, falls die angegebene Zahl größer als 88.0296919 ist.

FN
Typ: Dezimalfunktion
Beispiel: PRINT FN A(3)
Format: FN <name>(<zahl>)
Leistung: Liefert den Wert, der von der vom Benutzer definierten Funktion mit dem angegebenen Namen berechnet wird.
Fehlermeldungen: UNDEF'D FUNCTION, falls FN ausgeführt werden soll, ehe eine entsprechende DEF FN-Anweisung ausgeführt wurde.

FOR...TO...[STEP...]
Typ: Anweisung
Beispiel: 100 FOR V=2 TO 10 STEP 2
Format: FOR <zähler>=<startwert> TO <endwert>
[STEP <inkrement>]
Leistung: Führt alle Anweisungen zwischen dieser Anweisung und der zugehörigen NEXT-Anweisung zunächst so aus, daß der <zähler> den <startwert> hat, und addiert dann zu <zähler> gleich <startwert> das <inkrement> so lange hinzu, bis der <zähler> den <endwert> erreicht hat.

FRE
Typ: Ganzzahlige Funktion
Beispiel: FRE(0)
Format: FRE(<variable>)
Leistung: Stellt fest, wieviel Speicherkapazität im RAM noch verfügbar ist.

GET
Typ: Anweisung
Beispiel: GET X,X$
Format: GET <variablenliste>
Leistung: Liest jeweils ein einzelnes Zeichen aus dem Tastaturpuffer.

Fehlermeldungen: ?SYNTAX ERROR, falls eine Ziffer erwartet wird, aber ein Alphabetzeichen eingetippt wurde.

GET#
Typ: E/A-Anweisung
Beispiel: GET#1,C
Format: GET#<dateinummer>, <variablenliste>
Leistung: Liest jeweils ein einzelnes Zeichen vom spezifizierten Gerät bzw. von der spezifizierten Datei.

GOSUB
Typ: Anweisung
Beispiel: GOSUB 3400
Format: GOSUB <zeilennummer>
Leistung: Sprung innerhalb eines Programms zu der Anweisung mit der angegebenen Zeilennummer. Stößt das Programm in der Folge auf RETURN, wird mit der Anweisung unmittelbar hinter GOSUB fortgefahren.

GOTO
Typ: Anweisung
Beispiel: GOTO 870
Format: GOTO <zeilennummer>
Leistung: Sprung innerhalb eines Programms zur Anweisung mit der angegebenen Zeilennummer.

IF...THEN...
Typ: Anweisung
Beispiel 1: IF X<>4 THEN 340
Format 1: IF <ausdruck> THEN <zeilennummer>
Beispiel 2: IF X<>4 GOTO 340
Format 2: IF <ausdruck> GOTO <zeilennummer>
Beispiel 3: IF X<>4 THEN PRINT X:R=0
Leistung: Besitzt der Ausdruck den Wahrheitswert »wahr«, wird mit der Anweisung fortgefahren, die mit der hinter THEN bzw. GOTO angegebenen Zeilennummer versehen

ist, oder es werden alle hinter THEN noch folgenden Anweisungen ausgeführt. Hat der Ausdruck den Wahrheitswert »falsch«, dann wird mit der nächsten Anweisung im Programmtext fortgefahren.

INPUT
Typ: Anweisung
Beispiel: INPUT "ZAHL EINGEBEN:";N
Format: INPUT "<prompt>"; <variablenliste>
Leistung: Nimmt die vom Benutzer eingegebenen Zeichen bis zum nächsten RETURN entgegen und interpretiert diese Zeichen als Datenwerte und Zuordnung dieser Datenwerte zu den in der Liste angegebenen Variablen.
Fehlermeldungen: ?? bedeutet, daß eine weitere Eingabe erforderlich ist, d.h., daß noch nicht genügend Daten eingegeben worden sind, um alle Variablen der Liste mit Werten zu versehen. ?REDO FROM START bedeutet, daß man eine Zeichenreihe eingegeben hat, obwohl eine Zahl verlangt worden war, und daß man mit der Eingabe von vorne anfangen muß. ?EXTRA IGNORED bedeutet, daß mehr Werte eingegeben worden sind, als verlangt war, und daß die überzähligen Werte vom System ignoriert werden.

INPUT#
Typ: E/A-Anweisung
Beispiel: INPUT #1,A
Format: INPUT#<dateinummer>, <variablenliste>
Leistung: Nimmt Zeichen vom angegebenen Gerät bzw. der angegebenen Datei bis zum nächsten RETURN-Zeichen entgegen.

INT
Typ: Ganzzahlige Funktion
Beispiel: INT(–15.78)
Format: INT (<zahl>)

Leistung: Liefert den ganzzahligen Anteil einer Zahl. Ist die Zahl positiv, werden die Stellen nach dem Komma weggelassen, ist die Zahl negativ, dann ist das Ergebnis die nächstkleinere Zahl.

LEFT$
Typ: Stringfunktion
Beispiel: LEFT$(B$,7)
Format: LEFT$(<string>, <ganze zahl>
Leistung: Liefert aus einer Zeichenreihe die links stehenden Zeichen bis zu der durch die Zahl markierten Stelle.

LEN
Typ: Ganzzahlige Funktion
Beispiel: LEN("FRITZ") (Antwort: 5)
Format: LEN(<string>)
Leistung: Liefert die Anzahl der Zeichen der Zeichenreihe, also ihre Länge.

LET
Typ: Anweisung
Beispiel: LET X=(4+Y) 2
Format: [LET]<variable>=<ausdruck>
Leistung: Der Wert des Ausdrucks wird der Variable zugewiesen. Das Wort LET kann weggelassen werden.

LIST
Typ: Kommando
Beispiel: LIST 10-30
Format: LIST[[<erste zeile>]-[<letzte zeile>]]
Leistung: Auflistung der angegebenen Zeilen Ihres Programms.

LOAD
Typ: Kommando
Beispiel: LOAD "0:*",8,1

Format: LOAD["<dateiname>"][,<gerät>][,<adresse>]
Leistung: Liest eine Programmdatei von Magnetband oder Diskette in den Hauptspeicher. Dem Magnetbandgerät entspricht die Gerätenummer 1 (Voreinstellungswert), dem Diskettenlaufwerk die Gerätenummer 8.
Bei direkter Ausführung wird vor dem Laden des Programms CLR ausgeführt. Bei Verwendung innerhalb eines Programms führt das System das zu ladende Programm unmittelbar nach dem Laden sofort aus. Auf diese Weise können mehrere Programme miteinander verknüpft werden.
Wird innerhalb des Dateinamens das Zeichen * verwendet, dann wird die erste Datei herangezogen, auf die der restliche Dateiname paßt.
Das Programm wird ab Adresse 2048 in den Hauptsspeicher geladen, es sei denn, es ist 1 als Adresse angegeben; in diesem Fall wird das Programm an dieselbe Stelle geladen, an der es vor dem Sichern im Hauptspeicher gestanden ist.

LOG
Typ: Dezimalfunktion
Beispiel: LOG(34/32)
Format: LOG(<zahl>)
Leistung: Berechnet den natürlichen Logarithmus (zur Basis e) der Zahl.

MID$
Typ: Stringfunktion
Beispiel: PRINT MID ("ETAGE",2,3)
TAG
Format: MID$(<string>,<anfang>,<länge>)
Leistung: Liefert eine Teilzeichenreihe aus einem String, die mit dem angegebenen Zeichen (von links gezählt) beginnt und die angegebene Länge hat, wie aus dem angeführten Beispiel hervorgeht.

NEW
Typ: Kommando
Beispiel: NEW
Format: NEW
Leistung: Löscht das gerade im Hauptspeicher befindliche Programm.

NEXT
Typ: Anweisung
Beispiel: NEXT I,J,K
Format: NEXT [<zähler>] [<zähler>]...
Leistung: Bestimmt das Ende einer oder mehrerer FOR...-NEXT-Schleifen, wobei die Schleifen durch die Zählervariablen festgelegt werden. Wird keine Zählervariable angegeben, dann bezieht sich NEXT auf das letzte noch nicht abgeschlossene FOR.

NOT
Typ: Operator
Beispiel: IF NOT 3<4 THEN
Format: NOT <ausdruck>
Leistung: Liefert den Wahrheitswert »wahr«, wenn der Ausdruck den Wert »falsch« besitzt und liefert den Wahrheitswert »falsch«, wenn der Ausdruck den Wert »wahr« aufweist.

ON
Typ: Anweisung
Beispiel: ON K GOTO 450,670,300
Format: ON <variable> GOTO/GOSUB <zeilennummer>
[,<zeilennummer>]...
Leistung: Bewirkt die Fortsetzung der Programmausführung bei einer der angegebenen Zeilennummern, wenn die ganzzahlige Variable einen Wert gleich einer dieser Zeilennummern hat; andernfalls wird die Ausführung mit der nächsten Anweisung fortgesetzt.

OPEN
Typ: E/A-Anweisung
Beispiel: OPEN 1,8,15,"COMMAND"
Format: OPEN <dateinummer>[,<gerät>][,<adresse>]
[,"<dateiname>[,<typ>] [,<betriebsart>]"]
Leistung: Eröffnet einen Kanal für die Eingabe oder Ausgabe zu einem bestimmten Gerät. Voreinstellungswert für das Magnetband ist 1. Die Dateinummer muß dieselbe sein wie bei den entsprechenden Kommandos CLOSE, CMD, GET#, INPUT# und PRINT#.
Bei <adresse> bedeutet 1 das Eröffnen einer Datei auf einer Kassette, 2 bewirkt das Schreiben einer Bandendemarke bei Eingabe von CLOSE. Der <dateiname> darf aus maximal 16 Zeichen bestehen. Bei <typ> wird als Voreinstellungswert eine Programmdatei angenommen, für sequentielle Dateien ist SEQ und für relative REL anzugeben. Als <betriebsart> sind R für »Lesen« und W für »Schreiben« zulässig.

OR
Typ: Operator
Beispiel: A=4 OR B=2.
Format: <ausdruck> OR <ausdruck>
Leistung: Liefert den Wahrheitswert »wahr«, wenn einer der beiden Ausdrücke oder beide den Wert »wahr« haben. Weisen beide Ausdrücke den Wert »falsch« auf, ist das Ergebnis ebenfalls »falsch«.

PEEK
Typ: Ganzzahlige Funktion
Beispiel: PEEK(53280)
Format: PEEK(<speicherplatz>)
Leistung: Liefert die ganze Zahl zwischen 0 und 255), die gegenwärtig unter dem angegebenen Speicherplatz abgelegt ist.

POKE
Typ: Anweisung
Beispiel: POKE 53281,4
Format: POKE <speicherplatz>, <wert>
Leistung: Schreibt ein Byte (= acht Bit) in den angegebenen Speicherplatz.

POS
Typ: Ganzzahlige Funktion
Beispiel: POS(0)
Format: POS(<dummy>)
Leistung: Liefert die augenblickliche Position des Kursors (0 bis 39 bedeutet die erste, 40 bis 79 die zweite Zeile).

PRINT
Typ: Anweisung
Beispiel: PRINT "KASSENBELEG"
Format: PRINT[<variable>][,/;<variable>]...
Leistung: Bewirkt die Ausgabe der angegebenen Variablen von links nach rechts auf dem Bildschirm oder auf ein anderes Gerät. Bei Fehlen einer Ausgabeliste wird eine Leerzeile ausgegeben.
Durch die Interpunktionszeichen zwischen den Variablen wird der Abstand zwischen den auszugebenden Zeichen gesteuert. Ein Komma bedeutet, daß ab Beginn der nächsten Zone gedruckt werden soll (die 80-spaltige Ausgabezeile ist in acht Zonen mit je 10 Spalten unterteilt). Ein Semikolon fordert, daß unmittelbar hinter dem zuletzt ausgegebenen Element weitergedruckt werden soll. Leerzeichen zwischen Stringkonstanten oder Variablen haben dieselbe Wirkung wie ein Semikolon. Werden Leerzeichen an anderen Stellen verwendet, kann es Probleme geben. Nach Abschluß der Anweisung durch ein Komma oder durch ein Semikolon fährt die nächste PRINT-Anweisung dort fort, wo die letzte aufgehört hat, also in derselben Zeile und an der derselben Position.

PRINT#
Typ: E/A-Anweisung
Beispiel: PRINT#1 "MACH", R$; "PAUSE"
Format: PRINT#<dateinummer>[<variable>]
[,/;<variable>]...
Leistung: Identisch mit PRINT, abgesehen davon, daß die Ausgabe in die angegebene Datei gelenkt wird. Bei Dateien auf Magnetband haben Interpunktionszeichen grundsätzlich die Wirkung des Semikolon.
Falls die Liste nicht durch ein Interpunktionszeichen abgeschlossen ist, werden automatisch ein Wagenrücklauf und ein Zeilenvorschub eingefügt, anderfalls nicht.

READ
Typ: Anweisung
Beispiel: READ A,B,C
Format: READ <variable> [,<variable>]...
Leistung: Liest die Werte aus einer DATA-Anweisung in die angegebenen Variablen. Die Leseoperation beginnt mit dem ersten DATA-Element, das noch nicht gelesen worden ist, und endet, wenn allen Variablen ein Wert zugeordnet ist.
Fehlermeldungen: ?SYNTAX ERROR bedeutet, daß der Typ eines Datenelements nicht mit dem Typ der Variablen zusammenpaßt. ?OUT OF DATA meldet, daß nicht mehr genügend Daten für die angegebene Variablenliste vorhanden sind.

REM
Typ: Anweisung
Beispiel: REM SPESENABRECHNUNG
Format: REM [<bemerkung>]
Leistung: Diese Anweisung wird bei der Ausführung des Programms ignoriert; sie erscheint nur bei der Auflistung des Programms.

RESTORE
Typ: Anweisung
Beispiel: RESTORE
Format: RESTORE
Leistung: Setzt den Lesezeiger auf das erste Datenelement der DATA-Anweisung.

RETURN
Typ: Anweisung
Beispiel: RETURN
Format: RETURN
Leistung: Sprung auf die Anweisung unmittelbar nach der GOSUB-Anweisung, von der aus der Sprung in dieses Unterprogramm erfolgte.

RIGHT$
Typ: Stringfunktion
Beispiel: PRINT RIGHT$("UNTEN",3)
TEN
Format: RIGHT$(<string>,<ganze zahl>)
Leistung: Liefert aus einer Zeichenreihe eine Teilzeichenreihe, von rechts beginnend und in der durch die Zahl markierten Länge.

RND
Typ: Dezimalfunktion
Beispiel: RND(0)
Format: RND(<zahl>)
Leistung: Liefert eine dezimale Zufallszahl zwischen 0.0 und 1.0. Die angegebene <zahl> ist ohne Bedeutung, abgesehen von ihrem Vorzeichen. Positive Werte bewirken, daß eine sich irgendwann wiederholende Zahlenfolge geliefert wird. Null erzeugt eine sich wiederholende Folge von Zufallszahlen und ein negativer Wert bewirkt, daß bei jedem Aufruf mit einer anderen Zufallsfolge begonnen wird.

RUN
Typ: Kommando
Beispiel: RUN 35
Format: RUN[<zeilennummer>]
Leistung: Beginnt die Ausführung des aktuellen Programms, wobei mit der angegebenen Zeilennummer angefangen wird bzw. mit der ersten Zeile des Programms, wenn keine Zeilennummer angegeben ist.

SAVE
Typ: Kommando
Beispiel: SAVE "TEST.34",8
Format: SAVE ["<dateiname>"] [,<gerät>] [,<adresse>]
Leistung: Speichert das aktuelle Programm auf Magnetband oder auf Diskette. Der Diskette entspricht die Gerätenummer 8, dem Magnetband, die Gerätenummer 1, sie wird als Voreinstellungswert verwendet. Ist als Adresse 1 angegeben, dann wird das Programm beim Laden mit LOAD an dieselbe Stelle des Hauptspeichers gebracht, an der es zum Zeitpunkt der Sicherung steht. Die Angabe von 2 als Adresse bewirkt, daß das System eine Bandendmarke hinter das Programm setzt. Die Angabe von 3 als Adresse hat dieselbe Wirkung wie 1 und 2 zusammen.

SGN
Typ: Ganzzahlige Funktion
Beispiel: SGN(X)
Format: SGN(<zahl>)
Leistung: Liefert den Wert 1 bei positiven, den Wert –1 bei negativen Zahlen. Die Zahl Null ergibt den Wert 0.

SIN
Typ: Dezimalfunktion
Beispiel: SIN(X)
Format: SIN(<zahl>)
Leistung: Berechnet den Sinus der Zahl.

SPC
Typ: Ganzzahlige Funktion
Beispiel: SPC(0)
Format: SPC(<zahl>)
Leistung: Druckt die angegebene Anzahl von Zwischenräumen.

SQR
Typ: Dezimalfunktion
Beispiel: SQR(4)
Format: SQR(<zahl>)
Leistung: Berechnet die Quadratwurzel der Zahl.
Fehlermeldungen: ?ILLEGAL QUANTITY, falls die Zahl negativ ist.

STATUS
Typ: Ganzzahlige Funktion
Beispiel: STATUS
Format: STATUS
Leistung: Stellt fest, ob die letzte auf einer eröffneten Datei ausgeführte E/A-Operation schon abgeschlossen ist oder nicht.

STOP
Typ: Anweisung
Beispiel: STOP
Format: STOP
Leistung: Stoppt die Ausführung eines Programms und bewirkt den Übergang in die Betriebsart der direkten Ausführung.
Fehlermeldungen: ?BREAK IN LINE nnnn gibt an, in welcher Zeile die Ausführung des betreffenden Programms angehalten wurde.

STR$
Typ: Stringfunktion

Beispiel: PRINT STR$(1.567)
1.567
Format: STR$(<zahl>)
Leistung: Liefert die String-Darstellung der Zahl.

SYS
Typ: Anweisung
Beispiel: SYS(53280)
Format: SYS(<speicherplatz>)
Leistung: Sprung an den angegebenen Speicherplatz und Beginn der Ausführung eines Programms in Maschinensprache. Das entsprechende Maschinencodeprogrammstück muß mit einem RTS-Befehl (Rückkehr aus einem Unterprogramm) enden. Es wird dann mit der BASIC-Anweisung fortgefahren, die unmittelbar auf die SYS-Anweisung folgt.

TAB
Typ: Anweisung
Beispiel: PRINT "NAME"TAB(25)"BETRAG"
Format: TAB(<position>)
Leistung: Bewegt den Kursor auf die angegebene Position (zwischen 0 und 255).

TAN
Typ: Dezimalfunktion
Beispiel: TAN(.876)
Format: TAN(<zahl>)
Leistung: Berechnet den Tangens der Zahl.

TIME
Typ: Ganzzahlige Funktion
Beispiel: TI
Format: TI
Leistung: Gibt die Anzahl der Sekunden an, die seit dem Einschalten des Computers vergangen sind.

TIME$
Typ: Stringfunktion
Beispiel: TI$
Format: TI$
Leistung: Liefert eine Zeichenreihe aus sechs Zeichen, die die Zeit in der Form Stunden/Minuten/Sekunden angeben. Man kann durch Besetzung von TI$ die Zeit auch jederzeit selbst einstellen.

USR
Typ: Anweisung
Beispiel: USR(<zahl>)
Format: USR(X/3.4)
Leistung: Sprung in die Maschinensprachenroutine, deren Anfangsadresse die Speicherplätze 785 und 786 sind (in die man allerdings mit Hilfe von POKE bereits etwas hineingeschrieben haben muß). Die Zahl wird in den Gleitpunktakkumulator gebracht. Ist die Maschinensprachenroutine beendet, fährt das System mit der nächsten Anweisung nach der USR-Anweisung fort.

VAL
Typ: Dezimalfunktion
Beispiel: VAL("BEREIT")
Format: VAL(<string>)
Leistung: Liefert eine Zahl, die der Zeichenreihe entspricht. Ist das erste Zeichen der Zeichenreihe (Leerzeichen bleiben unberücksichtigt) weder ein Plus noch ein Minus noch eine Ziffer, dann ergibt sich Null als Wert.

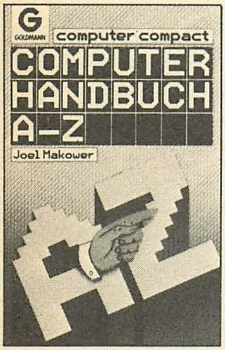
VERIFY
Typ: Kommando
Beispiel: VERIFY "TEST1",8
Format: VERIFY["<dateiname>"][,<gerät>]
Leistung: Überprüft, ob das bezeichnete Programm mit dem gegenwärtig im Hauptspeicher liegenden Programm

identisch ist. Voreinstellungswert für das Gerät ist 1 (Magnetbandgerät). Der Dateiname kann bei einer Magnetbanddatei weggelassen werden (in diesem Fall wird das nächste auf dem Magnetband gefundene Programm verwendet), muß bei einer Diskettendatei (Gerät 8) jedoch angegeben werden.
Fehlermeldungen: ?VERIFY ERROR entsteht, wenn irgendein Unterschied zwischen dem extern gespeicherten Programm und dem Programm im Hauptspeicher besteht.

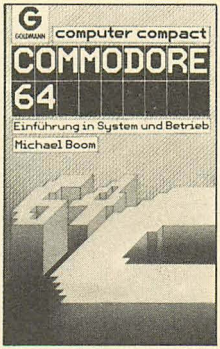
Goldmann computer compact

Kompetent. Kompakt. Verständlich und verläßlich.

Immer mehr wird der Personal- oder Homecomputer ein nicht mehr wegzudenkendes Hilfsmittel in Haus, Beruf, Schule, Industrie und Verwaltung.
Der Goldmann Verlag vermittelt dem Einsteiger und Fortgeschrittenen in seiner neuen Reihe das Verständnis für die wichtigsten Systeme und Programmiersprachen. Anwendungsbücher, für die jeder Computer-Interessent Verwendung hat!

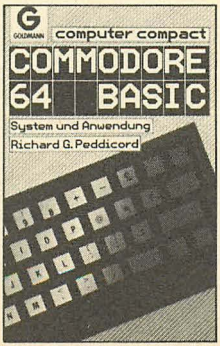


13113



13114

13115



13116

Joseph Reymann
CP / M
Einführung in das populärste Betriebssystem
Best.-Nr. 13117

Steven Manus
IBM PC
Einführung in System und Betrieb
Best.-Nr. 13118

In Vorbereitung:

Richard G. Peddicord
Commodore 64 Graphics
Einführung und Training
Best.-Nr. 13119
(Dezember '84)

Richard G. Peddicord
Apple Basic
System und Anwendung
Best.-Nr. 13120
(Dezember '84)

Richard G. Peddicord
Apple Graphics
Einführung und Training
Best.-Nr. 13121
(Januar '85)

Richard G. Peddicord
Logo
Einführung in die populärste Lernsprache
Best.-Nr. 13122
(Februar '85)

William Urschel
Wordstar
Einführung in das populärste Programm für Textverarbeitung
Best.-Nr. 13123
(März '85)

Carlton Shrum
Supercalc
Einführung und Training
Best.-Nr. 13124
(April '85)

COMMODORE 64 BASIC

Wer diesen vielseitigen Home Computer besitzt und beherrschen will, muß sich mit ihm perfekt »verständigen« lernen.

Dieses Buch ebnet dem Anfänger den Weg zur optimalen Nutzung der Leistungsfähigkeit des COMMODORE 64. Eine besonders effektive Starthilfe leisten die zahlreichen Testaufgaben.

- System und Anwendung
- Programmieren mit BASIC
- Einfache Operationen, praktische Übungen
- Peripheriegeräte

GOLDMANN VERLAG

ISB N 3-442-13116-2 DM +009.80

T 3-28-00